University of Alberta

Multi-zone Caching for IP Address Lookup

by

Ivan Leonidovitch Chvets

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Spring 2002

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-69695-2

Canada

University of Alberta

Library Release Form

Name of Author: Ivan Leonidovitch Chvets

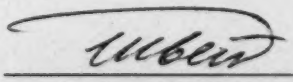
Title of Thesis: Multi-zone Caching for IP Address Lookup

Degree: Master of Science

Year this Degree Granted: 2002

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.



Ivan Leonidovitch Chvets
Department of Computing Science
University of Alberta
Edmonton, Alberta
Canada, T6G 2E8

Date: 22 March 2002

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Multi-zone Caching for IP Address Lookup** submitted by Ivan Leonidovitch Chvets in partial fulfillment of the requirements for the degree of **Master of Science**.

M. H. MacGregor

Michael H. MacGregor
Supervisor

José Nelson Amaral (Computing Science)

Xiaoling Sun (Electrical Engineering)
External Examiner

Date: March 8, 2002

To my parents Tatiana and Leonid Chvets,
to my sister Alena Chvets,
and also to Anna

Посвящается:

Моим родителям Татьяне и Леониду Швец,
моей сестре Алёне Швец,
и Анне

Abstract

Computer networks have become an integral part of every day life. Increase in both speed and size of computer networks puts a tremendous stress on the network interconnection devices – routers.

Network address lookup process, which must be performed by the router, becomes a major bottleneck when size and bandwidth of networks increase.

The goal of this thesis is to present a novel multi-zone caching technique which speeds up a routing table lookup.

A high degree of temporal and spatial locality was established by analyzing IP packet traces collected from large operational routers.

A complete multi-zone cache model was developed along with the optimal multi-zone cache design method. Trace-driven simulation confirmed the validity of multi-zone cache model and optimal multi-zone cache design method. The performance analysis showed that multi-zone IP cache significantly outperforms regular network address cache, thus, speeding up IP address lookup process.

Acknowledgements

I am very grateful to my supervisor Dr. Michael MacGregor for his patient guidance and support in all aspects of my research. Through his insightful discussions and suggestions, Dr. MacGregor helped me to understand the ingredients of the research process and maintained my interest in the work.

I would like to thank members of my examining committee, Dr. José Nelson Amaral and Dr. Xiaoling Sun, for their advice and corrections, all of which contributed to the successful completion of my thesis.

I would like to acknowledge resources and support provided by the Department of Computing Science of the University of Alberta.

I also would like to thank all faculty members, staff, fellow graduate students, and my friends who made my academic experience worthwhile and enjoyable.

I dedicate this work to my parents Tatiana and Leonid Shvets, to my sister Alena, and to Anna for their unconditional support throughout my academic and personal life and for numerous sacrifices these wonderful people had to make in order for me to see the success of my thesis.

Contents

1 Introduction 1

2 Overview of Locality and Caching 8

2.1 Locality 8

2.2 Caching 10

2.3 Summary 14

3 Locality and Caching in Computer Networks 15

3.1 Destination Address Locality: A Comparison of Caching Schemes . . 15

3.2 Improving Gateway Performance with a Routing Table Cache 18

3.3 Locality and Route Caches 21

3.4 Other Related Work 21

3.5 Summary 23

4 Locality in Internetwork Traffic 25

4.1 Locality in IP Traffic 25

4.1.1 Temporal Locality 26

4.1.2 Spatial Locality 27

4.1.3 Summary 29

4.2 Methodology 30

4.2.1 Experimental Model for Temporal Locality 30

4.2.2 Experimental Model for Spatial Locality 30

4.2.3 Experimental Data 31

4.2.4 Validation of Experimental Models 33

4.3 Experiments and Analysis 35

4.3.1 Temporal Locality 36
4.3.2 Spatial Locality 39
4.4 Summary 45

5 Multi-zone Cache for IP Address Lookup **46**
5.1 Cache for IP Address Lookup: Design Issues 46
5.1.1 Caching of IP Addresses 47
5.1.2 Related Aspects of Cache Design 47
5.1.3 Performance Analysis of IP Address Cache 49
5.1.4 Miss Ratio Prediction 50
5.2 Multi-zone Cache Design 52
5.2.1 Single-zone Cache Organization 52
5.2.2 Multi-zone Cache Organization 54
5.3 Multi-zone Cache Model 66
5.3.1 Access Time and Miss Ratio in Multi-zone Cache 66
5.3.2 Miss Ratio and Cache Size in Multi-zone Cache 71
5.3.3 Validation of Multi-zone Cache Model 80
5.4 Method for Optimal Multi-zone Cache Design 82
5.5 Summary 88

6 Performance Analysis of Multi-zone Cache **90**
6.1 Methodology 90
6.1.1 Experimental Model for Single-zone Cache 91
6.1.2 Experimental Model for Multi-zone Cache 91
6.1.3 Experimental Data 92
6.1.4 Validation of Experimental Models 93
6.2 Experiments and Analysis 93
6.3 Summary 102

7 Conclusions **104**
7.1 Locality in Internetwork Traffic 104
7.2 Multi-zone Cache for IP Address Lookup 105

7.3 Directions for Future Research 107

Bibliography **108**

A Data Collected on Traces **112**

B Performance of Multi-zone Cache: Simulation Results **115**

List of Tables

4.1 Traces of IP addresses collected at different sites. 32

4.2 Distribution of clusters. All traces. 40

5.1 Aggregation of IP addresses for the trace in Figure 5.3 (a). 57

5.2 Estimated parameters A and θ for all traces. 72

5.3 Estimated parameters A and θ for all traces; per-window fitting. . . . 75

6.1 Miss ratios for regular single-zone IP address cache with 292 cache slots (12288 bits of storage). 95

6.2 Initial parameters of two-zone cache designs. All traces. 96

6.3 Results produced by optimal design algorithm for each configuration. 97

List of Figures

1.1 The OSI reference model. 2
1.2 Internet Protocol Stack, OSI and TCP/IP reference models. 4
1.3 Generic router architecture. 6

2.1 Degree of associativity in cache. 12

4.1 Example of destination address traces. 26
4.2 Sample network. 28
4.3 Sample trace for validation of experimental models. 33
4.4 Validation of experimental model for temporal locality. 34
4.5 Validation of experimental model for spatial locality. 35
4.6 Number of references with interarrival times from 0.001 to 0.100 sec. Trace *A-1*. 36
4.7 Number of references with interarrival times from 0.001 to 0.100 sec. Trace *SDSC-1*. 37
4.8 Number of references with interarrival times from 0.001 to 0.100 sec. Trace *UofA*. 38
4.9 Cumulative percentage of references with interarrival times from 0.001 to 1.000 sec. All traces. 39
4.10 Clustering of *SDSC-1* trace. 41
4.11 Clustering of *SDSC-2* trace. 42
4.12 Clustering of *UofA* trace. 43
4.13 Cumulative percentage of clusters. All traces. 44

5.1 Single-zone IP address cache organization. 53
5.2 Examples of cache entry format. 54

5.3 IP address trace and routing table. 56
5.4 Multi-zone cache: caching common prefixes, fixed length search field. 58
5.5 Multi-zone cache: caching common prefixes, variable length search field. 60
5.6 Multi-zone IP address cache organization. 61
5.7 Multi-zone cache. 63
5.8 Fitting of footprint function $u(n) = A * n^{\frac{1}{\theta}}$. Trace *SDSC-1*. 72
5.9 Fitting of footprint function $u(n) = A * n^{\frac{1}{\theta}}$. Trace *SDSC-2*. 73
5.10 Fitting of footprint function $u(n) = A * n^{\frac{1}{\theta}}$. Trace *UofA*. 74
5.11 Fitting of footprint function $u(n) = A * n^{\frac{1}{\theta}}$ using multiple windows of references. Trace *SDSC-1*. 75
5.12 Fitting of footprint function $u(n) = A * n^{\frac{1}{\theta}}$ using multiple windows of references. Trace *SDSC-2*. 76
5.13 Fitting of footprint function $u(n) = A * n^{\frac{1}{\theta}}$ using multiple windows of references. Trace *UofA*. 77
5.14 Validation of miss ratio estimation technique. Trace *SDSC-1*. 79
5.15 Validation of miss ratio estimation technique. Trace *SDSC-2*. 80
5.16 Validation of miss ratio estimation technique. Trace *UofA*. 81
5.17 Optimization problem. 83
5.18 Algorithm for optimal multi-zone cache design. 85

6.1 Validation of multi-zone cache experimental model. 92
6.2 Miss ratio of single-zone cache with LRU. All traces. 94
6.3 Miss ratio of single-zone cache with OPT. All traces. 95
6.4 Simulated miss ratio of two-zone cache for S1[16:32] 98
6.5 Simulated miss ratio of two-zone cache for S2[20:32] 99
6.6 Simulated miss ratio of two-zone cache for U[23:32] 100
6.7 Simulated miss ratio of two-zone cache for U[12:32] 101

A.1 Number of references with interarrival times from 0.001 to 0.100 sec. Trace *A-2*. 112
A.2 Number of references with interarrival times from 0.001 to 0.100 sec. Trace *SDSC-2*. 113

A.3 Clustering of *A-1* trace. 113
A.4 Clustering of *A-2* trace. 114

B.1 Simulated miss ratio of two-zone cache for S1[14:32] 115
B.2 Simulated miss ratio of two-zone cache for S1[15:32] 116
B.3 Simulated miss ratio of two-zone cache for S1[17:32] 116
B.4 Simulated miss ratio of two-zone cache for S1[18:32] 117
B.5 Simulated miss ratio of two-zone cache for S1[19:32] 117
B.6 Simulated miss ratio of two-zone cache for S2[17:32] 118
B.7 Simulated miss ratio of two-zone cache for S2[18:32] 118
B.8 Simulated miss ratio of two-zone cache for S2[19:32] 119
B.9 Simulated miss ratio of two-zone cache for S2[21:32] 119
B.10 Simulated miss ratio of two-zone cache for S2[22:32] 120
B.11 Simulated miss ratio of two-zone cache for U[11:32] 120
B.12 Simulated miss ratio of two-zone cache for U[13:32] 121
B.13 Simulated miss ratio of two-zone cache for U[22:32] 121
B.14 Simulated miss ratio of two-zone cache for U[24:32] 122

Chapter 1

Introduction

Twentieth century saw a number of new technologies and inventions that have changed the way the modern world works. One of such technologies was a computer system. In early stages, computer systems were large, expensive and cumbersome, used mostly for scientific research and military purposes. It did not take long time for computer systems to evolve into compact, affordable and multi-purpose devices that affect many parts of every day life. One of the main tasks of a computer system is to process and store information in some way; however, this information can be accessed only by the user of the computer system it is stored on. In many cases people and organizations are willing to share the information they posses or to access the information provided by others. This lead to interconnection of physically separated computers into communication systems which are called *computer networks*.

Computer networks were first organized into Local Area Networks (LANs) that spanned separate offices and laboratories, campuses of universities and government agencies. Then these LANs were interconnected into Metropolitan Area Networks (MANs) that spanned cities. Later, cities, countries and even continents were interconnected with Wide Area Networks (WANs). Popularity of computer networks grew rapidly with time along with the number of different applications that use them. The system of world wide interconnection network became known as the Internet.

In order to reduce complexity of expanding old and building new networks and to enable compatibility of different types of computer networks, the logical communication model was introduced. This model is subdivided into number of *layers*, each one of these layers is built upon its predecessor. Each layer offers certain well-defined

set of services to layer higher in the hierarchy without providing information on how these services are actually implemented. The International Standards Organization (ISO) proposed a seven-layer model which was named the OSI (Open Systems Interconnection) Reference Model [Tanenbaum 96]. This model is presented in Figure 1.1.

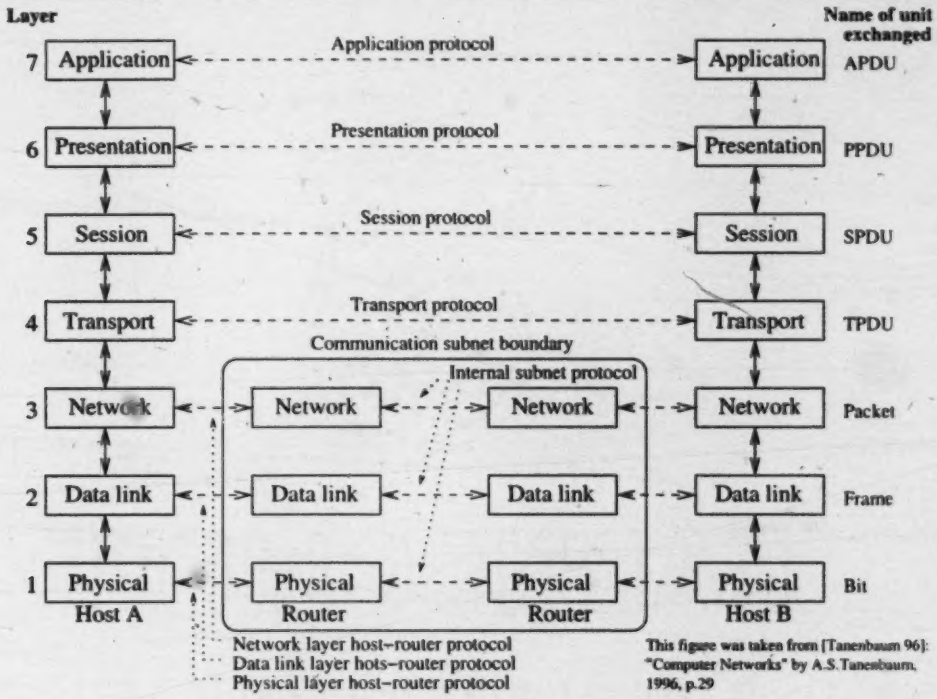


Figure 1.1: The OSI reference model.

The *physical layer* is concerned with transmitting raw bits over a communication channel. The design issues at this layer deal with the mechanical, electrical, and procedural interfaces, and the physical transmission medium.

Since physical layer is prone to errors due to noise and other physical factors, the main task of *data link layer* is to take a raw transmission facility and transform it into a communication line that appears free of transmission errors to the network layer.

The *network layer* is concerned with routing of packets, congestion control and overcoming differences between heterogeneous networks. Packets are forwarded by

network layer interconnection devices towards their ultimate destination. Routes, or paths, that packets are forwarded on, are selected depending on network address of each packet, current load of the network, type of service required by the packet, and other characteristics.

The main function of the *transport layer* is to accept data passed by the session layer, split it up into smaller segments, if required, pass them to the network layer, and ensure that all the pieces are re-assembled correctly at their ultimate destination. This layer is also responsible for establishing and maintaining logical connections across the network.

The *session layer* allows the establishment of sessions between different machines on the network. A session allows ordinary data transport, similar to the transport layer, but it also provides enhanced services used by some applications.

The *presentation layer* is concerned with the syntax of the information transmitted. This includes conversion of data between machines with different codes for representing characters or strings, data compression and encryption.

The *application layer* provides useful network applications to the end user, e.g., web browsers, electronic mail, remote login, and other.

Two approaches to network communication exist under the OSI model: circuit-switched networks and packet-switched networks. In circuit-switched networks, a connection is established between two hosts that are willing to communicate before any data can be sent over the network. In packet-switched networks, data is split into packets which are stored and then forwarded or routed by intermediate network nodes. Packet-switched technology is divided into three broad categories which are distinguished by size of geographical area they span, which was discussed earlier. Since this research is concentrated on large IP-based internetworks, the following discussion is focused on issues related to MANs, WANs and protocols used in these types of networks, especially at the network layer.

The OSI reference model is often compared to the TCP/IP reference model which is used as de-facto model of the Internet. At each layer there is a particular protocol which is responsible for the implementation of the services. These protocols are joined under the Internet Protocol Stack. Figure 1.2 presents the comparison of the OSI

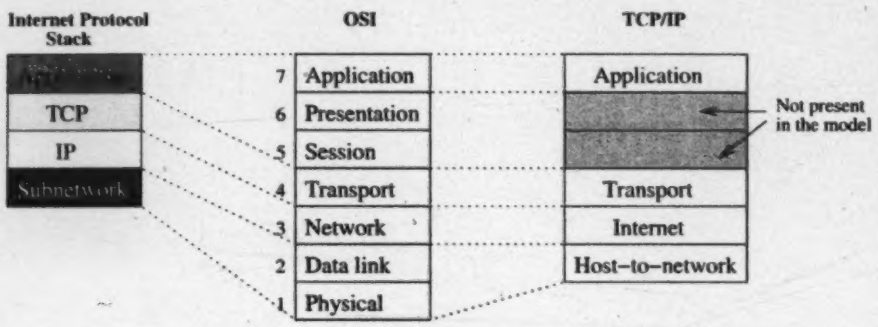


Figure 1.2: Internet Protocol Stack, OSI and TCP/IP reference models.

reference model (in the middle), the TCP/IP reference model (right), and Internet protocol stack (on the left). IP protocol (Internet Protocol) is responsible for the functionality at the network layer.

The specification of Internet Protocol is described in detail in [Postel 81]. IP version 6 is presented in [Deering *et al* 98]. Additional information on operation and extensions of IP can be found in [IETF]. According to the IP protocol addressing scheme, every host connected to the Internet receives a unique IP address. Each IP address is 32 bits in length and is represented by 4 octets written in a dotted-decimal notation, for example, `192.168.48.91`. Contemporary networks use classless inter-domain routing, which is described in detail in [Fuller *et al* 93]. It outlines subnetting and aggregation of IP network addresses which are used in transferring data from source to destination over the network.

In IP networks, subnetwork is a network sharing a particular *subnet address*. Subnetworks are networks arbitrarily segmented by a network administrator in order to provide a multilevel, hierarchical routing structure while shielding the subnetwork from the addressing complexity of attached networks. Sometimes subnetworks are called *subnets*. Address of a subnet is a portion of IP address that is specified as the subnetwork by the subnet mask. *Subnet mask* is a 32-bit address mask used to indicated the bits of an IP address that are being used for the subnet address. Usually, subnet masks are contiguous, thus they are represented by a single number

(mask length) which specifies the number os 'ones' in the mask. For example, a subnet address or a subnet prefix `192.168.128.0/17` has indication that its subnet mask has 17 ones in it (the notation `/17` at the end of IP address specifies the number of bits that identify the subnet prefix), so this mask can be viewed as `0xFFFF8000` or `255.255.128.0` in dotted decimal notation.

When an application running on one host is required to communicate to another host, it requests transport layer (managed by the TCP protocol) to establish a logical connection between hosts. When the connection is established the application passes data to the transport layer, which in its turn, passes it to the network layer (managed by IP protocol). Since the unit of transfer at the network layer is a packet, the data are split up into packets, called IP packets. Each packet is assigned a destination address – the address of the host where these packets should be transferred. After all IP packets arrive to that host they are reassembled by the TCP protocol and passed to the application layer, if required. When the source transmits packets, it determines one of its neighbours on the network that is closer on the path to destination and forwards packets to that node, which in turn, performs the same operation: forwards packets to the next network node on the path to the destination. The devices that are responsible for routing IP packets across the network are called *routers* (a generic router architecture is given in Figure 1.3).

Router, in itself, is a specialized computer system with a number of network interfaces. It is responsible for receiving, storing, and re-transmitting IP packets through its network interfaces or network cards according to a number of rules and policies specified at the configuration stage by the administrator or dynamically by routing protocols. Each router maintains a *routing table* in its memory which stores all known routes where each route consists of destination IP address, subnet mask, and other useful information. When an IP packet is received by the router, it performs lookup of this packet's destination IP address in its routing table. (A tree-based longest matching prefix routing table lookup technique is described in detail in [Sklower 91]). If a complete match or longest matching prefix is found, routing information is retrieved and the router forwards IP packets accordingly. If no match was found, it forwards IP packets over the default route. Since routing tables at major routers contain large

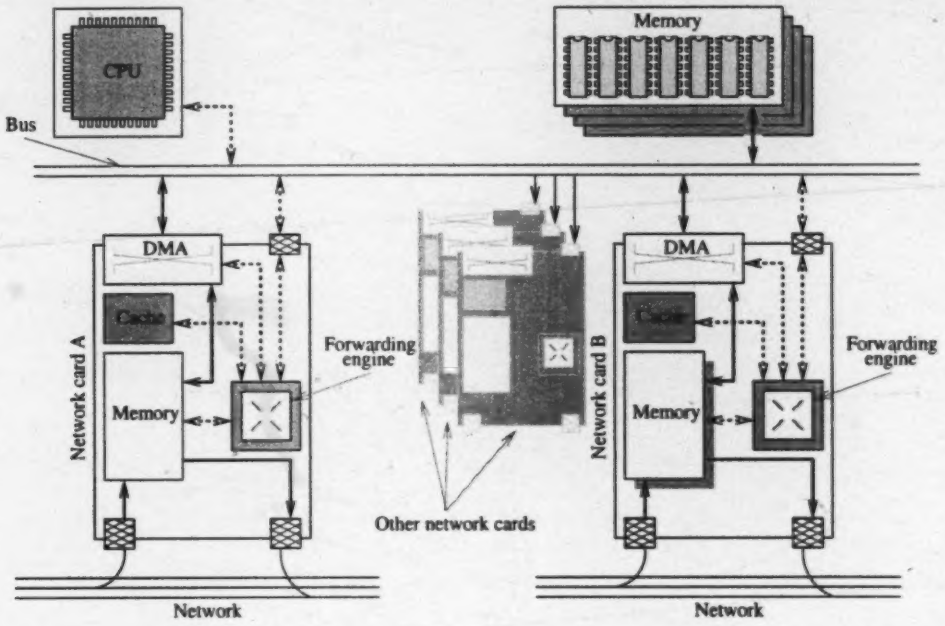


Figure 1.3: Generic router architecture.

number of entries, in the order of hundreds of thousands, and destination address lookup must be performed for every IP packet seen by the router, lookup process becomes a major bottleneck when traffic load on the network increases, especially at the core routers with high-speed network interfaces.

There were many methods introduced for improving performance of IP address lookup through new routing table lookup algorithms, introduction of new protocols, adjusting data structures to fit cache memory of the CPU. One of the methods proposed by the research community is to use caching of recently looked up IP addresses and the corresponding routing information, thus, avoiding costly routing table lookup.

The main goal of this research is to present a novel multi-zone caching technique for IP address lookup which takes into account spatial locality in a stream of references to IP addresses seen by the router. In addition, a multi-zone cache design method is presented.

Different types of locality – temporal and spatial – are identified in IP traffic using a set of IP address traces captured at different operational routers. A novel multi-zone caching technique is described along with optimal multi-zone cache design method. This method uses findings on spatial locality to predict the optimal cache configuration for given traffic conditions. The performance evaluation was performed by means of simulation of multi-zone cache with configurations estimated during the optimal design stage. To identify the benefits provided by an optimal multi-zone cache, its performance was compared to performance of regular IP address cache with LRU replacement policy. In addition, the optimal replacement policy was simulated.

Chapter 2 presents concept related to locality and caching in general. Chapter 3 summarizes previous research conducted in the area of network traffic locality and network address caching. Findings about locality in internetwork traffic are given in Chapter 4. Multi-zone caching technique and optimal multi-zone cached design are described in Chapter 5. Chapter 6 presents results of multi-zone cache performance evaluation. Conclusions and directions for the future research are summarized in Chapter 7.

Chapter 2

Overview of Locality and Caching

The research conducted in the area of memory and file referencing behaviour in computer systems has shown that these reference patterns follow non-uniform distribution in time and space. In other words, some memory locations or parts of a file are being accessed more frequently than others. To describe this behaviour the term "locality of reference" has been introduced. The locality of reference has been exploited in various ways to improve performance of computer systems. One technique is to use a small fast memory (cache) to store the most frequent accesses in order to speed up information retrieval. Research shows that this technique can be successfully used to speed up destination address lookup process in computer networks [Feldmeier 88], [Partridge 96], [Talbot *et al* 99], [Gulati 92]. This chapter presents an overview of the basic concepts of locality and caching.

2.1 Locality

Locality of reference has been studied in the context of memory reference behaviour [Bunt *et al* 84], file systems and virtual memories [Denning 70]. Generally, the principle of locality of reference can be described as follows [Hennessy *et al* 90]:

- If an item is referenced, it will tend to be referenced again in short period of time (*temporal locality*)
- If an item is referenced, neighbouring items will tend to be referenced soon (*spatial locality*)

There are identified two types of locality: *temporal locality* and *spatial locality*. Temporal locality suggests that the information last referenced has a high probability of being referenced again in the near future. For example, references performed by a program in a loop have a high degree of temporal locality, because data and instructions in the loop are being re-used. Spatial locality implies a high probability of referencing *neighbouring* regions of the region last referenced. For example, accessing elements of an array sequentially would produce a string of references with a high degree of spatial locality, because elements of the array are stored in memory at adjacent or neighbouring locations. The notion of neighbouring addresses for memory and file system addressing schemes is clear; however, for computer networks it requires some additional clarification. Addresses in computer networks can be considered in the same neighbourhood if they belong to some address group created by the implementation of some predefined addressing scheme for division of the network address space. These address groups are called subnets.

The terms *persistence* and *concentration* have also been used to characterize locality behaviour [Bunt *et al* 84]. Persistence refers to the tendency to repeat references to a single address. This is related to temporal locality. Concentration suggests a tendency for references to be limited to a small group of addresses within the whole address space. Concentration is similar to spatial locality.

Virtual memory systems successfully exploit locality concepts. The main idea behind virtual memory is that the combined size of the program is allowed to exceed the total amount of physical memory available in the system. The operating system keeps the most actively used pages of the program in main memory, and the rest of the program on secondary storage devices. Most programs at any given time reference only a limited number of their pages; in other words, they exhibit locality of reference, thus making virtual memory system an efficient solution for improving the performance of computer systems.

Cache memory systems in processors exploit locality of reference by storing recently accessed data or instructions. Information can be fetched or pre-fetched into a processor cache depending on the locality characteristics of the reference pattern. By storing frequently accessed information in a processor cache the average time needed

to access data and instructions is greatly reduced, since the time required to access the cache is significantly lower than the time needed to access main memory.

Locality of reference is also used in file systems to increase the efficiency of file access. This is similar to processor cache memory systems, only this time blocks of a file that resides on disk are cached in memory, thus reducing the time required for file access.

A number of methods of exploiting locality concepts for improving the performance of network interconnection devices have been proposed. These methods rely on caching references to network addresses, because a sequence of packets on the network can be viewed as a sequence of references. The research presented in this thesis is concerned with efficient caching techniques that exploit the locality of destination address references in network traffic.

2.2 Caching

Caching is widely used in modern computer systems to reduce access time to the data stored in main memory. The information that is likely to be currently in use is stored, or *cached* temporarily in a smaller, faster, and more expensive cache memory. Since access time to cache memory is 10% to 25% of the time needed to access main memory, cache memories significantly improve the performance of the computer system, when locality of reference is present [Smith 82]. (New computer systems have cache memory with access times of 1% to 3% of the time needed to access main memory).

Four aspects of optimal cache design have been identified [Smith 82]:

- Maximizing the probability (the *hit ratio*) of finding a reference's target in the cache (a *cache hit*); or, equivalently, minimizing the probability (the *miss ratio*) of **not** finding a reference's target in the cache (a *cache miss*).
- Minimizing the time required to access the cache (the *access time*).
- Minimizing the delay associated with a cache miss, i.e. combined time required to access cache and main memory to retrieve the required information.

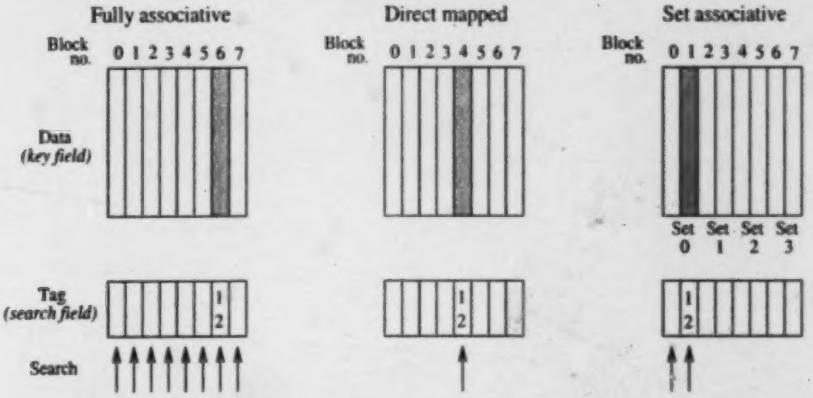
- Minimizing the overheads of updating main memory, maintaining cache consistency, etc.

Major issues in the design of cache memories are associativity, cache fetch method (demand or prefetch), placement and replacement algorithms (how and when information is stored in cache and what information is evicted from the cache to make space available for new data), size of each entry of the cache (line size), cache coherence schemes, main memory update mechanisms, and cache size (trade-offs between cache size and speed).

By the way items are being stored, caches are divided into three categories: *direct mapped*, *fully associative*, and *set associative*. In a direct mapped cache each item can be stored in only one place in the cache. The mapping is usually done modulo the number of entries in the cache. In a fully associative cache, an item can be placed into any location in the cache. Fully associative caches are normally more expensive and slower than other types of cache memories; however, the technology is improving, making fully associative caches attractive in some systems. The most popular cache organization method is set associative; under this scheme an item can be stored in a restricted set of places in the cache. Each set in such cache is fully associative. Set associative caches span the range from direct mapped caches where each cache entry is a separate set to fully associative cache where the whole cache is one set [Hennessy *et al* 90]. Figure 2.1 shows the comparison of storage and retrieval schemes for caches with different degrees of associativity. The set of rules which governs how items are placed in the cache is called the *placement policy*.

The most common cache fetch method is *demand fetching*. Under this method items are fetched into cache only when they are requested. This ensures correct information is fetched into cache, but it introduces the delay associated with accessing main storage. Although demand fetching cannot be removed completely from the process of fetching items into cache, some other technique can be used to ensure that information is in the cache before it is requested. The policy that fetches items before they are requested is called *prefetch*.

When a cache is full it is said to be in *steady state*. In steady state every cache miss will require a fetch to bring the *missed* information into the cache, and a step



In fully associative placement, the block for block–frame address 12 can appear in any of the 8 blocks; thus, all 8 tags must be searched. The desired data is found in cache block 6 in this example. In direct–mapped placement there is only one cache block where block 12 can be found. In set–associative placement, with 4 sets, memory block 12 must be in set 0 (12 mod 4); thus, the tags of cache blocks 0 and 1 are checked. In this case the data is found in cache block 1. Speed of cache access dictates that searching must be performed in parallel for fully associative and set–associative mappings.

This figure was taken from [Hennessy et al 90]: "Computer Architecture: A Quantitative Approach" by J.L.Hennessy and D.A.Patterson, 1990, p.410

Figure 2.1: Degree of associativity in cache.

to select the entry whose information will be replaced with newly fetched item. This requires the implementation of a replacement policy. Replacement policies can be divided into usage-based and non-usage-based. Usage-based policies select the entry for replacement based on its access frequency or usage. Examples of such polices are:

- LRU – Least Recently Used – selects the least recently accessed entry for replacement when required.
- LFU – Least Frequently Used – selects the entry that was accessed the least number of times.
- Working Set selects the entry which does not belong to the working set of the program currently running.

Polices such as FIFO (First In, First Out) and RAND (Random) can be grouped under the title of non-usage-based policies, since they select an entry for replacement

based on other factors suggested by the names of these methods. One particular replacement policy should be identified separately. MIN [Belady 66] is called OPT – *optimal* – for the purpose of this research. Under MIN, when a new item has to be stored in the cache, the entry selected is the one that will be accessed further in the future than any other entry currently in the cache. Since MIN replacement takes into account the whole trace of references, or, in other words, looks ahead in the future, it obviously, cannot be implemented in a real system. However, it serves as a very good measure for evaluation of other replacement techniques.

To take advantage of the spatial locality exhibited by programs, cache line size can be increased. For example, if a program accesses one word in main memory, the whole block is fetched into cache to fill up the cache line. Thus, neighbouring words are also brought into the cache. Since spatial locality suggests that neighbouring items are highly likely to be referenced in the near future, the next memory accesses performed by the program will not result in cache misses.

Cache coherence protocols are required to ensure correctness when multiple entities are able to access and change the contents of main memory and cache (and in some cases, multiple caches). There are many techniques introduced to solve the cache coherence problem and they are studied in the context of multi-processors systems. In relation to network address caches, coherence should be investigated when there is a possibility that routing information is changed in the routing table (which resides in main memory) and some network address cache entries become invalid. Since the process of routing table maintenance is usually centralized, a directory-based approach for cache coherence is a good candidate.

Another aspect of cache design is write policy. There are two major write policies: *write through* and *write back*. Under *write through*, updates are written both to cache and main memory. When *write back* (also known as *copy back*) is used updates are written to the cache and only when that entry is to be replaced is the updated information written to main memory [Hennessy *et al* 90].

The last issue in cache design is the cache size. When the cache is large the probability of finding information in the cache is high, and the hit ratio is high. However, cache size cannot be increased infinitely due to cost constraints, access time (larger

caches have lower access speeds), and physical space availability. The law of diminishing returns is also applicable to number of entries in the cache: after some optimal number of entries, each additional entry will start to bring a diminishing increase in performance. Larger cache requires bigger index decoders, multiplexers and other additional resources to support full associativity. As a result, in some cases it is economical to limit a cache to optimal or less than optimal size.

2.3 Summary

If a trace of references has non-uniform distribution over time it exhibits temporal locality. When references in a trace are limited to a subset of the address space, the trace exhibits spatial locality. The locality principle is effectively exploited in a number of systems such as virtual memory systems, and disk and processor caches. Efficient cache designs take locality into account to improve the performance of the whole system. Many aspects of cache design are well-studied and include fetch policies, placement and replacement methods, cache size, and main memory update policy. The principle of locality and aspects of cache design are directly applicable to data network and can be used in the design of efficient caching schemes for destination addresses.

Chapter 3

Locality and Caching in Computer Networks

Internetwork traffic consists of sequences of packets, where every packet has a destination address. As a result, internetwork traffic can be interpreted as a series of references, where the references have a non-uniform distribution over time and over the address space. Thus, it makes sense to investigate the nature of locality in internetwork traffic. Information about locality in network address traces can be exploited in the development of caching techniques which will improve the efficiency of processing at network nodes.

In this chapter previous work related to locality and caching in computer networks is described. These studies dealt with locality in local area networks, caching of network addresses at gateways, and terabit routers.

3.1 Destination Address Locality: A Comparison of Caching Schemes

Due to an increasing amount of traffic in networks more pressure is put on interconnection devices to direct this traffic at higher speeds between networks. This requires fast address lookup techniques to be implemented at routers and switches.

The research presented in [Jain 90] is concerned with the problem of address recognition in bridges and suggests that the locality in network traffic, if present, can be exploited in designing caching schemes that could speed up address lookup at inter-

connection devices. [Jain 90] chooses the trace driven approach for investigation of traffic locality. Traces of destination addresses collected at a local area network were analyzed for locality. In particular, this work presents an investigation of the applicability of three well known locality models to network traffic: IRM (independent reference model), LRU (least recently used), and WS (working set model) [Jain 90].

The independent reference model suggests that all references in the trace of addresses are independent. That is, that the last reference was to address k does not provide any information about the next address. This model assumes that the trace of addresses does not have either temporal or spatial locality [Jain 90]. However, as this study showed, the distribution of references to destination addresses was not uniform. The cumulative frequency of references as a function of fraction of distinct addresses in the trace chosen for this study showed that 50% of all frames were destined to 4% of all destinations and 90% of the frames were destined to 17% of the destinations. This clearly shows that destination references exhibit a strong *concentration* or, in other words, have high degree of *spatial locality.*

Under the working set model addresses referenced in the last W references are highly likely to be referenced again. The interval W is called the *working set window size*, and the number of distinct references in the interval is called the *working set size.* If there is temporal locality in the trace, then the working set size would be small in comparison to the working set window size [Jain 90]. As the study showed, the destination reference pattern exhibited a high degree of temporal locality. There were 65 distinct destinations referenced (working set size) for a working set window size of 500. However, for a working set window size less than 50, there was no apparent temporal locality. This is explained by the fact that the working set size is very close to the working set window size.

The LRU stack model assumes that the probability of referencing a destination address is a decreasing function of the time since this address was last referenced [Jain 90]. The study found that the top 100 stack positions or 20% of the total stack positions accounted for 98% of all frames in the trace.

The evaluation of popular cache replacement algorithms is also presented in [Jain 90]. The algorithms evaluated were: least recently used (LRU), first in first out (FIFO),

random (RAND), and the optimal algorithm OPT. The metrics used to evaluate these algorithms were miss probability, interfault distance, and normalized search time. It should be noted that OPT algorithm assumes knowledge of the whole trace, thus it cannot be implemented in a real system; however, it provides a theoretical limit for performance and can be used in evaluating other techniques.

It was found that for small caches LRU, FIFO, and RAND did not differ significantly in terms of miss probability for the trace of references chosen. However, OPT performs better by a factor of two [Jain 90].

Interfault distance analysis showed that LRU was close to optimal algorithm, and FIFO and RAND perform equally poorly. It was also suggested that the LRU algorithm could be used as a replacement algorithm for large network address caches due to its close-to-optimal performance.

Another question raised was concerned with the optimal cache size, and the suggested way for addressing this question was the analysis of the third metric, normalized search time. The search time for a table of n entries in main memory can be assumed to be $1 + \log_2(n)$. The search time for a cache of c entries is $1 + \log_2(c)$. In case of a cache hit it will take $1 + \log_2(c)$ to find the address in the cache; in case of a cache miss the total time to lookup address will increase to $1 + \log_2(c) + 1 + \log_2(n)$. Normalized search time t is defined as:

$$t = \frac{\textit{Search time with cache}}{\textit{Search time without cache}}$$

$$= \frac{(1-p) * (1 + \log_2(c)) + p * (1 + \log_2(c) + 1 + \log_2(n))}{1 + \log_2(n)},$$

where p is the probability of a cache miss.

Results showed that LRU replacement performs better than FIFO and RAND; however, it is still outperformed by OPT. The optimal size of the cache was found to be 64 entries for the trace investigated. In addition, it was found that using a very small cache increases search time by 20% compared to a system without a cache and that the gain decreased as cache size increased. This leads to the conclusion that caching can be harmful in some cases.

The research conducted by Jain was focused on an investigation of destination

traffic locality in local area networks and possible ways to exploit locality to speed up the forwarding process in bridges. Four different replacement algorithms were compared.

Since [Jain 90] is limited to locality and caching in local area networks it cannot be directly compared with the research presented in this thesis which deals with locality and caching in large internetworks. However, the concepts and methods presented in [Jain 90] can be applied to internetwork traffic with some adjustments.

3.2 Improving Gateway Performance with a Routing Table Cache

In order for computers on local area networks to communicate with the rest of the world, the networks must be interconnected. The interconnection device is called a gateway. The major purpose of a gateway is to direct traffic among the networks it connects and to ensure that packets are forwarded correctly according to their destination address. If a gateway is not directly connected to the destination network to which the packet should be forwarded, it sends the packet to another gateway which lies on the shortest path from the source of the packet to its destination. The address of the next gateway is the *next-hop* address. Gateways use the destination address of a received packet as an index to a *routing table* which contains a collection of `<Destination Address/Next Hop>` pairs, thus determining on which interface the packet should be retransmitted. A routing table is maintained by the gateway [Feldmeier 88].

The research conducted by Feldmeier [Feldmeier 88] was focused on the effectiveness of a routing table cache at the gateway and the organization of the cache. Since routing table lookup is a time consuming process, the use of a routing table cache which stores recently seen destination addresses should decrease the time required to forward a packet if there is locality in the packet address stream. The trace-driven approach was chosen for this research. A trace was gathered for an operating gateway by recording all routing table references during a period of normal operation

[Feldmeier 88]. This trace was then applied to a simulated gateway containing a destination address cache. A fully associative cache was simulated because it has the best hit ratio for a given cache size. In addition, the use of a fully associative cache removes any measurement bias because of specific network addresses and makes the results general for any set of network addresses with similar characteristics [Feldmeier 88].

Two replacement strategies were selected for this study: LRU (Least Recently Used) and FIFO (First In, First Out). Also, two addressing schemes were examined during the study: *flat addressing*, which uses the entire internet address space for the cache simulation, and *hierarchical addressing*, which uses only the network field of the internet address. The cache was pre-loaded to measure steady-state performance and to avoid cache misses caused by cache initialization. In addition, two fetching schemes were investigated: *destination fetching* which suggests storing only the destination address in the cache and *destination/source fetching* which suggests storing both the destination and source addresses of each packet.

Experiments were conducted using flat addressing and varying the replacement policies and fetching techniques. One of the characteristics of cache performance was identified as the relationship between cache hit ratio and the probability of access:

$$f_h(i) = \sum_{j=1}^{i} p_j$$

$f_h(i)$ is the cache hit ratio as a function of i, the number of slots in the cache; and p_j is the probability of a packet address being the j^{th} previous reference.

The graph of cache hit rate versus cache size showed that even for small cache sizes the probability of a cache hit increases rapidly. With as few as 9 slots, the hit ratio is above 0.9 (or 90%). The results suggested that LRU replacement is very effective for caching packet destination addresses.

Another characteristic is the relationship between cache hit ratio and the number of packets between cache misses:

$$f_m(i) = \frac{1}{1 - f_h(i)}$$

$f_m(i)$ is the number of packets between cache misses as a function of i, the number of cache slots.

It was found that destination/source fetching with LRU replacement performed best, followed closely by destination fetching with LRU replacement. Performance of all FIFO caches was relatively poor. In addition, increasing the number of cache slots showed a diminishing increase in performance for large caches.

The experiments were repeated for hierarchical addressing. In this case it was assumed that the first three bytes of the internet address referred to the network and last byte referred to the host. The results obtained showed that destination/source LRU and destination LRU performed much better compared to FIFO schemes. The cache hit ratio was 0.9 (90%) with as few as 7 cache slots.

A cost analysis was performed, and its results showed that lookup time could be effectively reduced to 43% of its intial value for flat addressing and LRU replacement. The optimal cache size for this scheme was found to be 16 slots. For hierarchical addressing the performance of the LRU cache was even better, since dropping the last byte of the internet address reduced the number of distinct addresses that the cache must handle. The optimal cache size was 51 slots and the average lookup time was 39% of that for memory-based table lookup. A more efficient cache design could further reduce the average lookup time. Since cache was implemented in software, the cost analysis was concerned with a number of instructions required to lookup network address in the table with and without the forwarding table cache.

Feldmeier attempted to evaluate the performance of a gateway that was equipped with a routing table cache to exploit traffic locality. The results obtained illustrate that there is a significant degree of locality in the network traffic, in particular, there is a significant degree of temporal locality. In addition, it showed the effectiveness of caching addresses to speed up packet forwarding by reducing lookup times by as much as 65%.

Since Feldmeier's research was concerned with locality and caching at the network layer it is closely related to the research presented in this thesis and has been useful in developing the methodology for the analysis of results.

3.3 Locality and Route Caches

The research presented in [Partridge 96] dealt with locality and route caches in backbone routers. It extends the research discussed earlier in [Feldmeier 88]. Traces used in the experiments were longer and were collected at a major backbone gateway so that traffic intensity was 600 times higher and the network was 100 times bigger than in [Feldmeier 88]. The cache was not pre-loaded as in [Feldmeier 88] and the number of cache slots in the cache was varied between 1 and 10000. As in [Feldmeier 88] cache was fully-associative and was managed by LRU replacement policy.

Two types of measurements were taken. First, the average hit rate was measured. The results showed that for a cache size of 500 the average hit rate was 0.50 (or 50%). For a cache size of 5000, the average hit ratio reached 0.93 (93%). This hit rate is much lower than in [Feldmeier 88] given the number of cache slots, but it should be noted that the cache is not pre-loaded in this case which results in a large number of misses during cache initialization. The second set of measurements included measuring hit rate in a setting where the cache was reset every 100000 packets. The hit ratios for this scheme were slightly lower, but still performance was very high: for a cache with a reset interval of 100000 references and 5000 slots the average hit rate was 0.91 (91%). These results confirmed the presence of significant locality in the network traffic and that caching could be very helpful in increasing the processing speed of gateways.

This research provided updated results for locality and route caches and re-iterated the fact that locality in network traffic could be exploited in caching IP addresses to speed up the packet forwarding process. It also stated that even a small cache could improve performance significantly.

3.4 Other Related Work

There are a number of other studies that are related to the research presented in this thesis. These studies include [Gulati 92] which deals with locality and caching in local area networks; caching of IP addresses [Talbot *et al* 99]; internetwork traf-

fic locality [Claffy 94]; network address lookup using CAMs [McAuley *et al* 93]; efficient layer-four packet caching [Xu *et al* 00]; compaction of routing table information for efficient routing table lookups [Gupta *et al* 98]; and others that investigate use of different data structures for routing table organization, alternative search algorithms for routing table lookup and improvements of IP address lookup using general purpose CPU cache [Chiueh *et al* 99 a], [Chiueh *et al* 99 b], [Lampson *et al* 98], [Degermark *et al* 97], [Waldvogel *et al* 97], [Varghese *et al* 98], [Nilsson *et al* 98].

The goal for the work presented in [Gulati 92] was to determine the extent to which network traffic locality could be exploited to improve the performance of a data link layer interconnection device. This research identified four characteristics of locality for destination and source addresses: persistence, address reuse, concentration, and reference density. In addition, host-pair locality was identified as important characteristic. Real traffic traces collected on a LAN were used to measure the above characteristics and it was found that LAN traffic had low persistence, or, in other words, a low degree of temporal locality. In addition, it was identified that traffic exhibited high re-use of both source and destination addresses, and host-pair locality as well. High concentration and reference density were found for both the source and destination addresses. The analysis also showed that the source address of the current and last few frames can be more helpful in predicting future references than the destination address. Based on findings about traffic locality in LANs, a forwarding-table cache was proposed to reduce the mean forwarding-table lookup time.

[Gulati 92] described different replacement and fetch policies that could be implemented for the forwarding-table cache. Trace driven simulation was used to investigate different replacement policies such as LRU, LFU, FIFO, RAND, and MIN. LRU was found to be very efficient in reducing the miss ratio in comparison to other schemes. The miss ratios for LRU were 20% higher than those for MIN. This performance gap shows that there exists room for improvement as far as replacement policies are concerned. Fetch policies were investigated under LRU replacement with cache size varying from 1 to 75 slots. The source fetch policy performed best for small cache sizes up to 13 slots. Beyond this cache size destination fetch policy performed better. An analysis was conducted to determine the improvements in mean

lookup time that could be achieved by using a forwarding-table cache. It was found that with LRU replacement and destination fetch, a forwarding-table cache reduced lookup time by 70%, assuming the cache is three times faster than the forwarding table.

The research described in [Talbot *et al* 99] investigated how IP address caching could improve destination address lookup in terabit routers. The authors suggested exploiting the temporal locality in IP traffic to improve IP address lookup rates. They also used trace driven simulation for this research and collected a number of large traces from different sites in US backbone network. The number of cache slots was varied from 4000 to 256000. In comparison to other studies these numbers are very large; however, the authors were mostly concerned with the speed and cost of the cache which they implemented using DRAM chips. The cache hit ratios were very high depending on the trace, ranging from 0.95 (95%) to 0.87 (87%) for a cache with 4000 slots.

3.5 Summary

This chapter summarized relevant research conducted in the area of locality in network traffic and caching for improving network address lookup. [Jain 90] presented a discussion on locality and evaluation of different replacement algorithms used in caching of network addresses. Jain also pointed out that there is significant temporal and spatial locality in network traffic in LANs. The research presented in [Feldmeier 88] investigated the effectiveness of a routing table cache in network address lookups. Flat and hierarchical addressing was taken into account due to the different caching requirements for these two approaches. It was found that even a small cache was able to reduce address lookup times by 65%. Partridge [Partridge 96] repeated the experiments conducted by Feldmeier [Feldmeier 88] under heavier load and on a network with a much larger number of hosts. This research restated the suggestion that caching is in fact an effective way to improve the performance of an interconnection device; the hit rates achieved were found to be around 90%. There

were also other studies conducted in the area of improving network address lookup rates using different techniques.

In summary, previous research has shown the existence of locality in internetwork traffic and examined a number of ways this locality can be exploited in order to improve the performance of network interconnection devices. The research presented in this thesis deals with aspects of locality in internetwork traffic, in particular, IP traffic, and investigates new cache organization and design techniques that exploit locality to accelerate network address lookups.

Chapter 4

Locality in Internetwork Traffic

Several previous studies have identified the presence of locality in traces of network traffic. In addition, a number of methods have been suggested for exploiting locality to improve the performance of the interconnection devices (switches and routers). These methods include locality based caching of network addresses, and locality-aware data structures and algorithms. However, few studies have been concerned with locality in large internetworks whose main addressing scheme is regulated by IP (Internet Protocol). This chapter presents findings about temporal and spatial locality of reference in large IP-based internetworks.

First, locality concepts are adapted to locality in IP traffic. Then experimental methods used to identify locality are presented along with the experimental data. The analysis of temporal and spatial locality is given and some conclusions are presented.

4.1 Locality in IP Traffic

As stated previously, network traffic can be viewed as a sequence of packets where each packet has its own destination address. This sequence of addresses possesses locality of reference. As a result, terms such as temporal and spatial locality can be adapted to the discussion of locality in IP traffic.

Two different types of locality - temporal and spatial - can easily be identified. Temporal locality means that there is a high probability of referencing the same address within a short period of time. That is, an address that has been referenced

recently is more likely to be referenced again than one that has not been seen for awhile. This is due, in part, to traffic passing across the network in "trains" of packets [Jain 90]. Spatial locality means that there is a high probability of referencing addresses in the same numerical range, or network (*neighborhood*). The region or *neighborhood* may be a group of addresses, a subnet, or a group of subnets.

Arrival Time:	(a)	(b)	(c)	(d)
0.01	192.168.205.76	192.168.205.76	192.168.45.125	192.168.45.125
0.02	192.168.205.76	192.168.205.98	192.168.45.129	192.168.45.127
0.03	192.168.205.76	192.168.205.98	192.168.205.98	192.168.45.126
0.04	192.168.205.76	192.168.205.76	192.168.45.130	192.168.205.76
0.05	192.168.205.76	192.168.205.98	192.168.201.19	192.168.205.98
0.06	192.168.205.76	192.168.205.98	192.658.45.126	192.168.45.125
0.07	192.168.205.76	192.168.205.76	192.168.201.45	192.168.45.127
0.08	192.168.205.76	192.168.205.98	192.168.205.98	192.168.45.126
0.09	192.168.205.76	192.168.205.98	192.168.45.125	192.168.205.76
0.10	192.168.205.76	192.168.205.76	192.168.45.126	192.168.205.98
0.11	192.168.205.76	192.168.205.98	192.168.45.130	192.168.45.125
0.12	192.168.205.76	192.168.205.98	192.168.45.129	192.168.45.127
0.13	192.168.205.76	192.168.205.76	192.168.201.45	192.168.45.126
0.14	192.168.205.76	192.168.205.98	192.168.201.19	192.168.205.76
0.15	192.168.205.76	192.168.205.98	192.168.205.98	192.168.205.98

Figure 4.1: Example of destination address traces.

4.1.1 Temporal Locality

Internetwork traffic often consists of sequences of packets that share a destination address; in IPv4, each packet has a 32-bit destination address [Maufer 99]. For example, the sequences of packets depicted in Figure 4.1 (a) and (b) have a high degree of temporal locality because they both have a high probability of referencing the same IP address in a short period of time. The sequence in Figure 4.1 (a) consists of 15 references to only one IP address `192.168.205.76` with an interarrival time of 0.01 seconds. This means that 100% of all packets in the trace destined to the same address have interarrival time of 0.01 seconds. However, for the trace in Figure 4.1 (b) the number of packets with interarrival time of 0.01 seconds is smaller because this sequence of packets consists of two different flows of packets destined to IP addresses `192.168.205.76` and `192.168.205.98` respectively and packets with the same IP addresses are non-uniformly distributed in the trace. 38% of all packets with common destination address have interarrival time of 0.01 seconds, 31% have interarrival time

of 0.02 seconds, and the remaining 31% have interarrival time of 0.03 seconds. This means that the trace in Figure 4.1 (a) has a higher degree of temporal locality than the trace in Figure 4.1 (b).

In comparison to the traces in Figures 4.1 (a) and (b), the sequences in Figures 4.1 (c) and (d) have relatively poor temporal locality. Trace in Figure 4.1 (c) consists of 8 flows with each flow destined to a different IP address. The distribution of packets with common IP address according to their interarrival times for this trace is as follows: 20% have interarrival time of 0.04, another 20% have interarrival time of 0.05, 10% have interarrival time 0.06, 20% have interarrival time of 0.07, 10% have interarrival time of 0.08, 10% have interarrival time of 0.09, 10% have interarrival time of 0.10. Trace depicted in Figure 4.1 (d) has 5 flows where 50% of packets, that share the same destination address, have interarrival times of 0.05 and another 50% have interarrival time of 0.06.

Based on the above discussion, the definition for temporal locality in IP traffic is stated as follows: a trace of references to IP addresses has high temporal locality when a large portion of the repeated references have a short interarrival time, i.e. a trace has a large probability of re-referencing the same IP address in a short period of time.

4.1.2 Spatial Locality

To define spatial locality for IP traffic, first the term *neighborhood* should be explained in relation to IP addressing. A neighborhood is the collection of addresses which are close physically or numerically to some given address. For example, the neighborhood for the address 7 is a collection of addresses {5, 6, _ , 8, 9}. Neighborhood addresses can easily be observed in memory or on disk. For network addresses a neighborhood is a set of hosts physically close to each other. Then there is no neighborhood relation implied by the numerical proximity of network addresses, because hosts can be in the same physical location, but on different local area networks that use different addressing schemes. In the case of IP addressing, addresses can be considered in the same group (neighborhood or cluster) if they have a common prefix of some predefined length. Generally, this prefix will be an address of a physical or

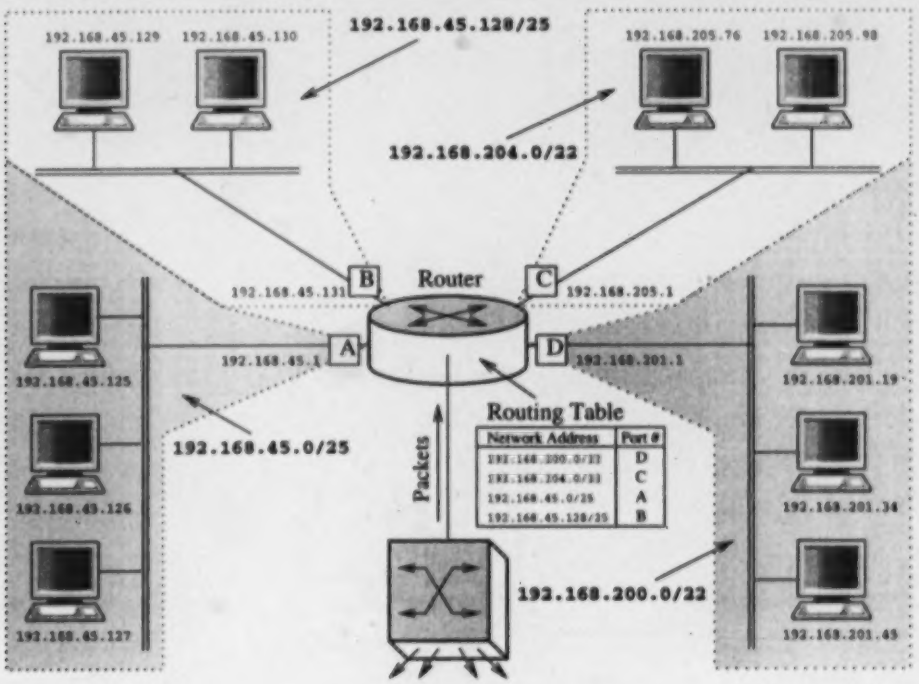


Figure 4.2: Sample network.

virtual network on which the hosts are located.

Consider sample network presented in Figure 4.2. There are 4 subnets which were created using IP addressing [Fuller *et al* 93]. There is a router in the middle of the network which connects these four networks. The routing table of the router is also shown. Each of these subnetworks can be identified as a neighborhood or a cluster of IP addresses. Now consider the traces presented in Figure 4.1. Traces (a) and (b) have high spatial locality, because they contain references to only one neighbourhood that is aggregated under the network address of `192.168.204.0/22` . All packets in these traces will be routed, according to the routing table shown, to the subnet connected to the router via port `#C`.

In contrast, the trace presented in Figure 4.1 (c) has very low spatial locality, because it contains references to all 4 subnetworks of the sample network given in Figure 4.2: `192.168.200.0/22`, `192.168.204.0/22`, `192.168.45.0/25`, `192.168.45.128/25`.

As a result, the packets will be routed to all 4 subnets that are connected to the router. It contains references to a large number of distinct subnets or clusters in comparison to the total number of subnets available (in this case only 4 subnets are available to be referenced). It should be noted that this particular trace also has very low temporal locality. Thus, it is possible for the sequence of references to have low locality in both dimensions – temporal and spatial.

However, even though the trace presented in Figure 4.1 (d) has low temporal locality it has higher spatial locality than the trace depicted in Figure 4.1 (c). This trace has references to only two subnets out of the four possible: `192.168.204.0/22`, `192.168.45.0/25`. According to the routing table some packets in this trace will be routed to subnet connected via port `#C` and other packets to subnet connected via port `#A`. This shows that there is a possibility for traces to have low temporal and high spatial locality.

To summarize, spatial locality for IP traffic is defined as follows: a trace of references to IP addresses has high spatial locality if it references a limited number of the available subnets, i.e. a trace has a high probability of referencing addresses from the same subnet.

4.1.3 Summary

There is the possibility of traffic locality of both types – temporal and spatial – in internetwork traffic, due to the nature of IP routing, addressing and aggregation schemes. However, one must carefully investigate actual IP traffic in order to implement efficient locality-based caching and routing techniques. Since currently available models do not replicate real IP traffic very well, trace driven simulation is used in this work.

4.2 Methodology

To analyze different types of locality in IP traffic, several methods were developed based on well-known sampling techniques described in the literature [Claffy *et al* 93], [Rueda *et al* 96]. This section describes the methods used to gather information on temporal and spatial locality in internetwork traffic along with the characteristics of the experimental data.

4.2.1 Experimental Model for Temporal Locality

To assess the degree of temporal locality in a stream of internetwork addresses the distribution of interarrival times of packets was analyzed. The method used to collect the necessary data was based on computing the number of IP destination addresses repeated within a given time interval. In other words, when characterizing temporal locality the intervals between repeated references to the same address were taken into account rather than just the interarrival time between two successive packets on a link. The count of the number of repeated references was calculated for a range of values of the interval. This was done by scanning the packet trace for the destination address and arrival time of each packet. The interarrival time was computed for each repeated address discovered in the trace, and the number of arrivals in the time bin for that interarrival time was incremented. After the whole trace had been scanned, each time bin contained a count of repeated references that had the corresponding interarrival time.

4.2.2 Experimental Model for Spatial Locality

For analysis of spatial locality a clustering algorithm was developed. It organizes the address space that was accessed by the whole trace into clusters in order to identify spatial behaviour of the reference stream. The clusters produced by this algorithm are groups of IP addresses with a common prefix. Thus, each cluster can be thought of as representing a subnet with an address mask equal in length to the number

of bits in the common prefix of the addresses in this cluster. For example, IP addresses 192.168.45.67, 192.168.45.101 and 192.168.45.7 can be considered as lying in the same cluster with prefix 192.168.45.0, or equivalently in the subnet 192.168.45.0/24. The input data for the clustering algorithm is the set of unique IP addresses found in a trace.

The algorithm works as follows. First, the starting mask length is selected; the default is a starting mask of zero. Then the set of unique IP addresses is scanned and the mask is applied to every one of them. For example, the address 192.168.45.67 under a mask of length 24 becomes 192.168.45.0 because each of the four decimal numbers in the address represents an 8-bit quantity. The number of IP addresses with the same prefix after applying the mask is counted, and these addresses are recognized as forming a cluster. If there are more than a predefined number (this predefined number is called *cluster size*; for this research cluster size was set at 32) of addresses in a cluster it is split further in the next iteration. If the number of addresses placed in a cluster is less than the cluster size, this cluster is considered complete and all addresses belonging to it are marked as *clustered*. The mask length is then incremented and the procedure is repeated, skipping all IP addresses marked as clustered. The process of declaring clusters when they are less than the limiting value in size is appropriate because it starts with a mask length of zero which potentially covers all addresses in the trace, and iterate towards longer, more specific masks. The number of clusters discovered can be interpreted as the number of subnets present in the original trace.

4.2.3 Experimental Data

The experimental data used in this research consisted of traces – sequences of IP packet headers – collected at real gateway routers at various sites [WAND], [NLANR], [UA]. The characteristics of the traces discussed in this research are presented in Table 4.1.

The two traces, *A-1* and *A-2*, are collections of long GPS-synchronized IP headers and corresponding timestamps that were captured with a DAG2 system at the

University of Auckland Internet uplink by the WAND research group in November 1999 [WAND]. The tap was installed on an OC3 link (155.52 Mbps) carrying a number of Classical-IP-over-ATM, LANE and POTS services. The trace contains all Classical-IP headers of a single VPI/VCI pair, which connects the university to the local service provider. A maximum 2 Mbps peak packet rate was set in each direction. These traces (*A-1* and *A-2*) have been sanitized by mapping the addresses onto `10.X.X.X` network to preserve privacy. This makes the analysis of spatial locality in these traces impossible. However, temporal locality should be unaffected by the sanitation process [WAND]. Only ICMP, TCP and UDP packets appear in the trace. For UDP packets and IP fragments all user payload is zeroed. There was no information provided on the type or characteristics of the router that handles the described connection.

Trace	A-1 Auckland II short	A-2 Auckland II long	SDSC-1 SDSC short	SDSC-2 SDSC long	UofA U of A
Number of IP packets	3626938	30270778	3619341	31518464	999990
Number of unique IP addresses	1622	68167	28474	130163	5797
Run length (estimated)	03:11:30 or 11489 sec	38:29:11 or 138551 sec	00:19:58 or 1198 sec	02:43:38 or 9818 sec	00:01:11 or 71 sec
Packets/second	316 pps	218 pps	3021 pps	3210 pps	141,000 pps

Table 4.1: Traces of IP addresses collected at different sites.

The next two traces, *SDSC-1* and *SDSC-2*, are collections of selected fields of IP packet headers and the timestamps captured at the San Diego Supercomputer Center commodity connection in 1995. The traces were not sanitized which makes them ideal for investigation of both temporal and spatial locality. As with the Auckland traces the router characteristics were not given.

The last trace, *UofA*, was collected in May 2001 at the University of Alberta, at the major connection between the university and the local ISP, Telus Communications Inc.. The trace is a collection of IP packet headers with timestamps obtained by using `tcpdump`. The router responsible for connection between the university and ISP is a Cisco 7507 with R5000 RISC processor at 200Mhz, 256MB of DRAM, 2MB of SRAM for packet buffers and 512KB of L2 cache. It has 2Gbps backplane and has seven modular slots. The interface the trace was captured on is 100BaseFx,

Address trace

Arrival Time	*IP Address*
0.001	192.168.118.17
0.002	192.168.118.91
0.003	192.168.12.5
0.004	10.134.93.112
0.005	10.45.67.98
0.006	192.168.45.142
0.007	10.182.17.81
0.008	192.168.118.3
0.009	192.168.202.123
0.010	10.18.125.12
0.011	10.134.118.121
0.012	192.168.202.71
0.013	192.168.118.17
0.014	10.134.93.112
0.015	192.168.118.91
0.016	192.168.118.3
0.017	192.168.202.123
0.018	192.168.45.142
0.019	10.45.67.98
0.020	10.18.125.12
0.021	192.168.118.3
0.022	10.18.125.12
0.023	192.168.202.71
0.024	192.168.12.5
0.025	10.134.93.112
0.026	192.168.12.5
0.027	10.182.17.81
0.028	192.168.118.17
0.029	192.168.45.142
0.030	10.18.125.12
0.031	10.134.118.121
0.032	192.168.12.5

Spatial locality

1 cluster with mask length 9 bits

10.45.67.98
10.18.125.12

2 clusters with mask length 11 bits

10.182.17.81

10.134.118.121
10.134.93.112

1 cluster with mask length 17 bits

192.168.202.71
192.168.202.123

1 cluster with mask length 18 bits

192.168.12.5
192.168.45.142

2 clusters with mask length 26 bits

192.168.118.91

192.168.118.3
192.168.118.17

Total of 7 clusters is created for the given trace

Temporal locality

Interarrival Time	*Number of References*
0.002 sec	2
0.005 sec	1
0.006 sec	1
0.008 sec	3
0.010 sec	2
0.011 sec	3
0.012 sec	2
0.013 sec	1
0.014 sec	1
0.015 sec	1
0.020 sec	2
0.021 sec	1

Unique IP Addresses:

192.168.118.17
192.168.118.91
192.168.12.5
10.134.93.112
10.45.67.98
192.168.45.142
10.182.17.81
192.168.118.3
192.168.202.123
10.18.125.12
10.134.118.121
192.168.202.71

Figure 4.3: Sample trace for validation of experimental models.

which is contained in Versatile Interface Processor module which keeps its own copy of forwarding table and does distributed switching. Trace *UofA* was not sanitized to preserve the spatial locality data.

Also, it should be noted that the above traces have different traffic intensity. *UofA* trace has the highest throughput of packets per second in comparison to other traces. Possible reason for this is that the IP traffic intensity on the interface where *SDSC-1* and *SDSC-2* traces were collected is lower than on the interface at the University of Alberta. Detailed comparison could not be done due to the lack of information on the router and interface characteristics at San Diego Supercomputer Center.

4.2.4 Validation of Experimental Models

The experimental methods described above were implemented using `C` programming language and its standard libraries. To make sure that temporal and spatial locality

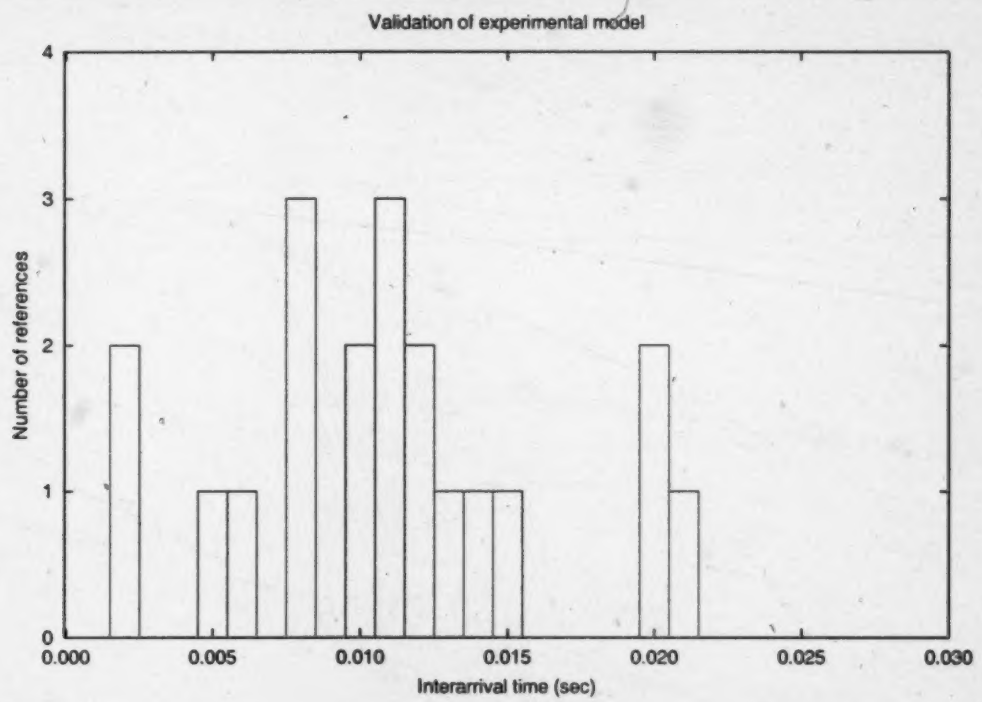


Figure 4.4: Validation of experimental model for temporal locality.

data collected are valid these models are checked for correctness. Usual approach to this task is to run the experimental models using data sets with known characteristics. Figure 4.3 presents IP address trace with arrival time of the corresponding packets. Temporal and spatial locality characteristics are shown for the trace.

The experimental models were run with IP address trace presented in Figure 4.3. Temporal and spatial locality data collected by these experiments match the expected results and are presented in Figures 4.4 and 4.5. Additional experiments were performed with known distributions of IP addresses in time and space as input traces. From the analysis of the data collected it was concluded that experimental models correctly capture temporal and spatial behaviour of IP traces, and can be used in analysis of temporal and spatial locality in internetwork traffic.

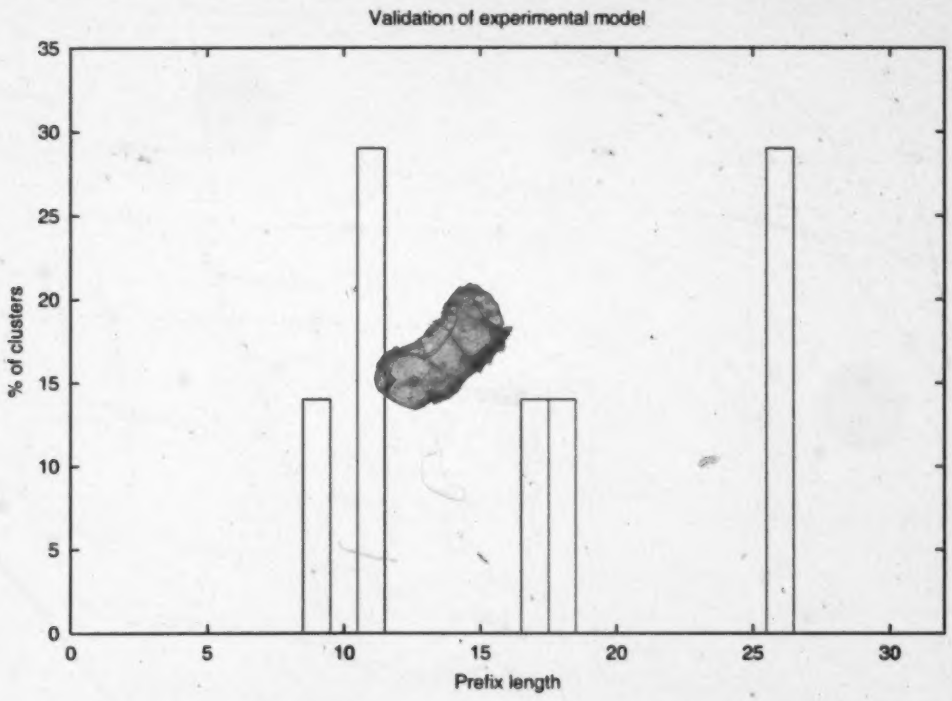


Figure 4.5: Validation of experimental model for spatial locality.

4.3 Experiments and Analysis

Measurements of temporal and spatial locality were conducted using the methods and IP address traces described above. For temporal locality experiments the width of each time bin was selected to be 0.001 seconds to ensure more accurate capture of the temporal behaviour of the traces. The range of interarrival times of repeated references to the same IP address was chosen to be between 0 and 5 seconds, which resulted in a total of 5000 time bins. Interarrival times of repeated references to the same address greater than 5 seconds were discarded. These values amounted to 1.7% of the total number of references in the *A-1* trace, 1.4% in trace *A-2*, 2.5% in *SDSC-1*, 3.4% in *SDSC-2*, and 0.0005% in the *UofA* trace. For the spatial locality experiments, the cluster size limit was selected to be 32, i.e. if 32 or less IP addresses form a cluster it is considered to be a complete cluster and it is not split during the

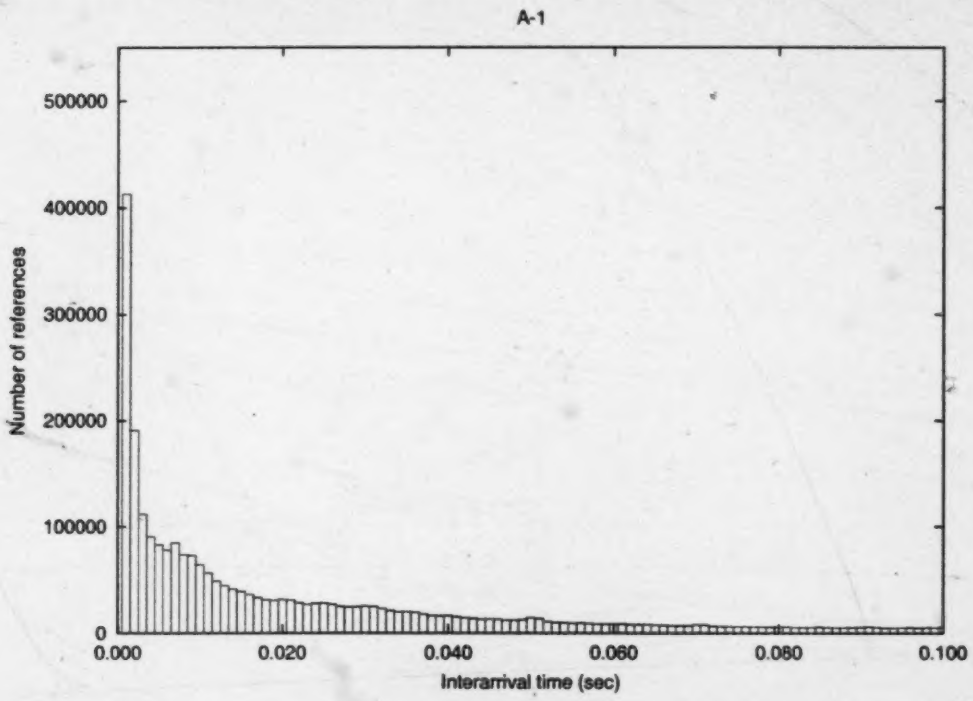


Figure 4.6: Number of references with interarrival times from 0.001 to 0.100 sec. Trace *A-1*.

later iterations of the clustering algorithm.

Data collected on temporal locality are presented in Figures 4.6 through 4.9, and Figures A.1 and A.2. Spatial locality data are presented in Table 4.2, Figures 4.10 through 4.13 and Figures A.3 and A.4. Traces *A-1* and *A-2* were not included in the discussion of spatial locality, because their spatial locality was destroyed by the sanitation process.

4.3.1 Temporal Locality

Figures 4.6 through 4.8 and Figures A.1 and A.2 show the distributions of interarrival times of repeated references. Figure 4.9 presents a comparison of the cumulative distributions for all traces.

Figures 4.6 through 4.9, A.1, and A.2 show that the distribution of references to

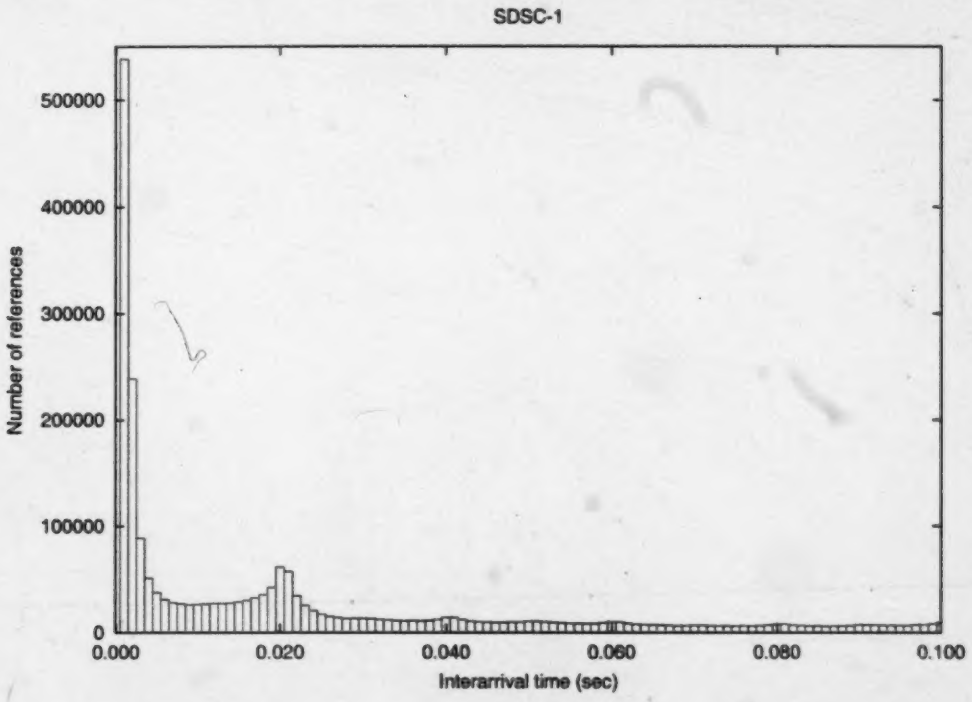


Figure 4.7: Number of references with interarrival times from 0.001 to 0.100 sec. Trace *SDSC-1*.

the same IP address is non-uniform for all traces, and that a large number of repeated references have very short interarrival times. Approximately 80% of all references have interarrival times less than 0.200 seconds. This percentage is highest for the fastest connection *UofA*. This suggests a high degree of temporal locality, because there is a high probability of referencing the same IP address within a short period of time.

For trace *A-1* 63.9% of all references have interarrival times less than 0.050 seconds and 72.5% occur in less than 0.100 seconds. *SDSC-1* has similar distribution where 53.7% of all references have interarrival times a less than 0.050 seconds and 64.2% are less than 0.100 seconds. The *UofA* trace has even larger proportions of references with short interarrival times: 77.1% are less than 0.050 seconds and 84.7% are less than 0.100 seconds. In addition, the percentage of references with interarrival times of 0.001 seconds or less is overwhelming in all traces: *A-1* - 11.6%, *A-2* - 13.6%,

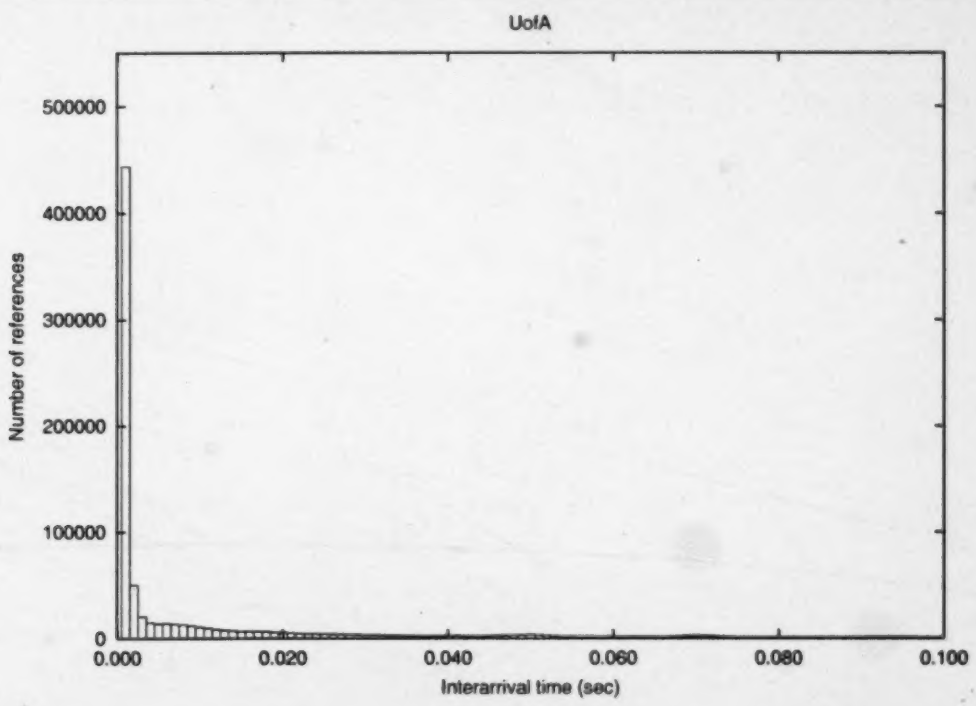


Figure 4.8: Number of references with interarrival times from 0.001 to 0.100 sec. Trace *UofA*.

SDSC-1 - 15.4%, *SDSC-2* - 20.1%, *UofA* - 44.8%. This means that a large number of packets with the same destination IP address follow each other in the shortest time interval recorded. Thus, the probability of referencing the same destination address in a short period of time is very high. This demonstrates that there is a high degree of temporal locality in these traces.

The comparison of cumulative distributions of interarrival times for all traces (Figure 4.9) shows that large number of all references have very short interarrival times in comparison to the longest recorded. The cumulative distributions in traces *A-1* and *A-2* have interarrival times less than 0.250 seconds for 85.6% and 85.8% of all references respectively. Similarly, traces *SDSC-1* and *SDSC-2* have interarrival times of less than 0.250 seconds for 81.2% and 82.7% of all packets. As for the *UofA* trace, 93.4% of all references have interarrival times less than 0.250 seconds. This provides

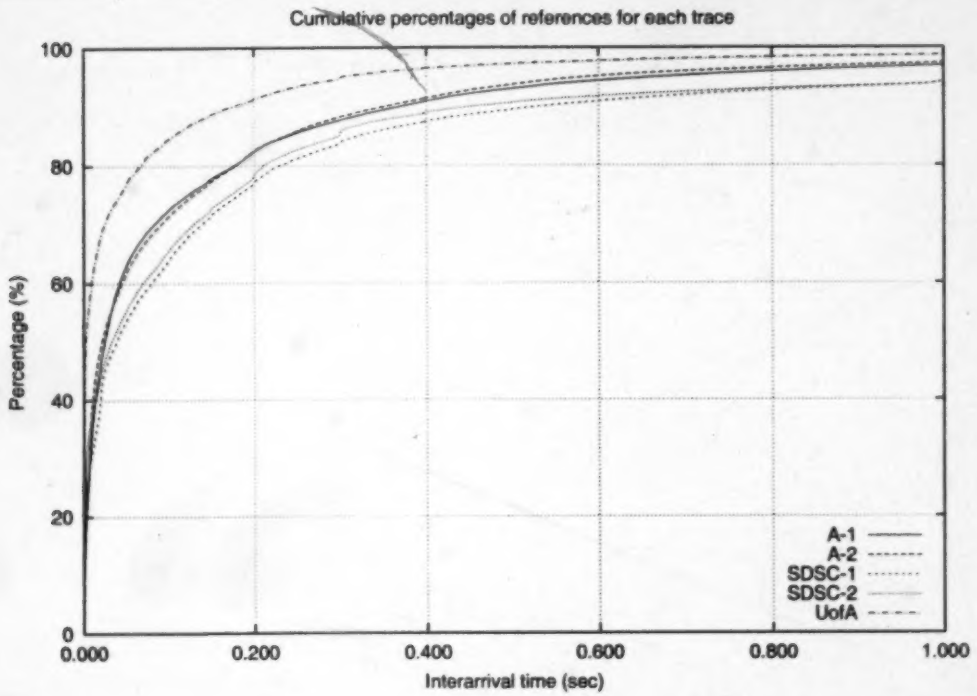


Figure 4.9: Cumulative percentage of references with interarrival times from 0.001 to 1.000 sec. All traces.

additional support for the claim that there is a high degree of temporal locality in traces of IP addresses.

The analysis of temporal locality in these traces shows that they have a high degree of temporal locality, thus making it possible to apply the methods and concepts developed for locality of reference in operating systems to the investigation of temporal locality in IP traffic.

4.3.2 Spatial Locality

Table 4.2 and Figures 4.10 through 4.12 present the distributions of clusters in the traces *SDSC-1*, *SDSC-2*, *UofA* according to the prefix length of a cluster. Figure 4.13 shows the cumulative distribution of clusters for all traces. In this discussion the

Prefix length	A-1	A-2	SDSC-1	SDSC-2	UofA
0-2	0%	0%	0%	0%	0%
3	0%	0%	0.06%	0%	0.31%
4	0%	0%	0.13%	0%	0.31%
5	0%	0%	0.19%	0.04%	0.62%
6	0%	0%	0.44%	0.14%	2.15%
7	0%	0%	0.38%	0.19%	2.77%
8	0%	0%	0.69%	0.21%	0.92%
9	0%	0%	2.14%	0.19%	4.31%
10	0%	0%	3.97%	0.37%	11.38%
11	0%	0%	3.91%	1.62%	16.31%
12	0%	0%	4.98%	3.06%	16.62%
13	0%	0%	13.93%	3.54%	8.62%
14	0%	0%	20.10%	5.12%	4.62%
15	0%	0%	18.21%	9.58%	2.46%
16	0%	0%	9.39%	16.51%	0.92%
17	0%	0%	4.16%	21.12%	2.15%
18	6.17%	0%	4.03%	14.06%	1.85%
19	16.05%	0%	4.28%	7.27%	2.15%
20	17.28%	0%	2.39%	5.01%	1.85%
21	4.94%	0%	2.02%	3.63%	2.77%
22	14.82%	0.78%	1.32%	1.94%	7.69%
23	14.82%	1.41%	0.63%	1.02%	5.85%
24	14.82%	2.10%	0.95%	1.02%	1.23%
25	8.64%	1.55%	0.69%	0.88%	0.62%
26	2.47%	1.73%	0.38%	0.98%	0.92%
27	0%	92.43%	0.63%	2.50%	0.62%
28-32	0%	0%	0%	0%	0%

Table 4.2: Distribution of clusters. All traces.

same definition of cluster is used as in the description of the clustering algorithm.

Figures A.3 and A.4 present the distributions of clusters in traces *A-1* and *A-2* respectively. When these traces were originally collected, the addresses were mapped onto `10.X.X.X/8` subnet. This is a common procedure used to preserve privacy when cataloging traces. The clustering results presented in Figures A.3 show that addresses in the trace *A-1* are concentrated in clusters with mask lengths between 18 and 26 bits. Figure A.4 also shows that in trace *A-2* the clusters are concentrated in one particular region of the address space. The addresses for this trace are concentrated in clusters which have mask lengths between 22 and 27 bits; approximately 92% of all clusters have a mask length of 27 bits. Table 4.2 presents more detailed distributions of prefix lengths of clusters. Distributions of clusters in traces *A-1* and *A-2* significantly differ from those of non-sanitized traces. Since mapping of IP addresses in these traces was performed based on location of the IP address in a trace and not on prefix of the address, such mapping assigns IP addresses that have same prefixes different IP

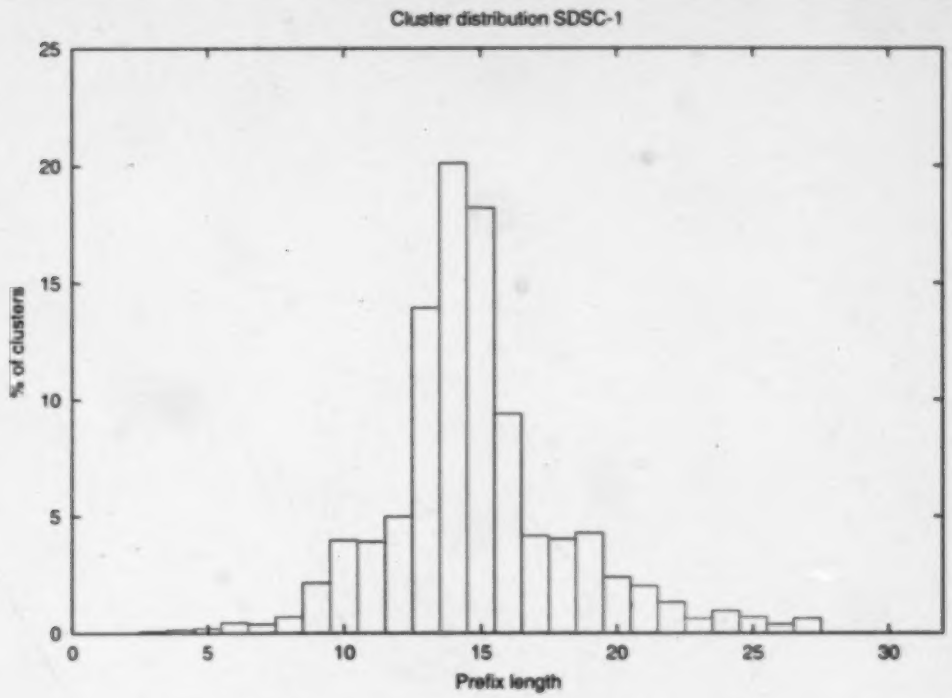


Figure 4.10: Clustering of *SDSC-1* trace.

addresses in 10.X.X.X network that do not share same prefix. As a result, packets which are destined to IP addresses on the same subnet are assigned IP addresses that are not related in space and are on different subnets in the address space represented by 10.X.X.X.

Thus, it is concluded that sanitation process performed on these traces destroyed much of a spatial locality data which makes these two traces (*A-1*, *A-2*) not useful in analysis of spatial locality in IP address stream. Only three traces are considered for further discussion *SDSC-1*, *SDSC-2* and *UofA*, since they were not sanitized or mapped in any way.

The cluster distributions for traces *SDSC-1* and *SDSC-2*, shown in Figures 4.10 and 4.11, follow non-uniform bell-shaped patterns. For the trace *SDSC-1*, 79.1% of all clusters have prefix lengths between 12 and 19 bits, and for the trace *SDSC-2* 78.7% of all clusters have prefix lengths between 14 and 20 bits. There are no clusters

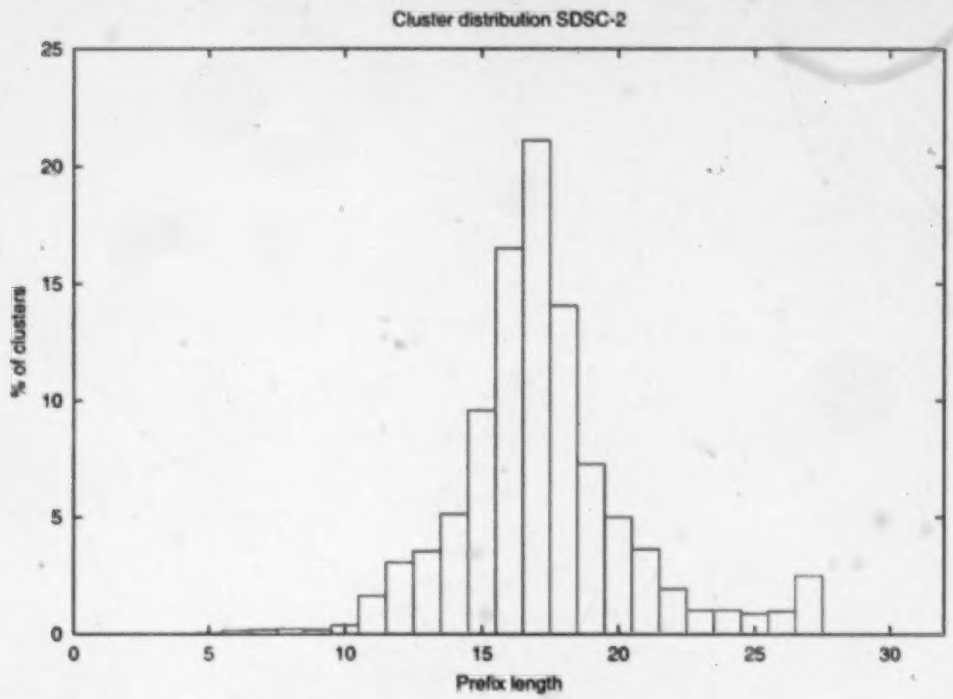


Figure 4.11: Clustering of *SDSC-2* trace.

with prefixes of 0 to 2 or 27 to 32 bits for trace *SDSC-1*, and for trace *SDSC-2* there are no clusters with prefixes of 0 to 4 bits, nor clusters with prefixes over 28 bits in length.

The distribution of clusters in the *UofA* trace (Figure 4.12) follows a different distribution. It has two peaks at prefix lengths values of 12 and 22 bits. Overall, 61.9% of all clusters have prefixes of 9 to 14 bits in length and 13.5% have prefix lengths of 22 and 23 bits. There are no clusters whose prefix lengths are 0 to 2 bits, nor clusters with prefixes over 28 bits in length.

The above data suggest that a large number of subnets have prefixes of specific lengths (12 to 19 bits for *SDSC-1*, 14 to 20 bits for *SDSC-2*, and 9 to 14 bits for *UofA*) that are responsible for over 75% of the references in IP traces, which supports the claim that there is a high degree of spatial locality in IP traffic.

In the cluster distribution for the *SDSC-1* trace (Figure 4.10), 86.9% of all clusters

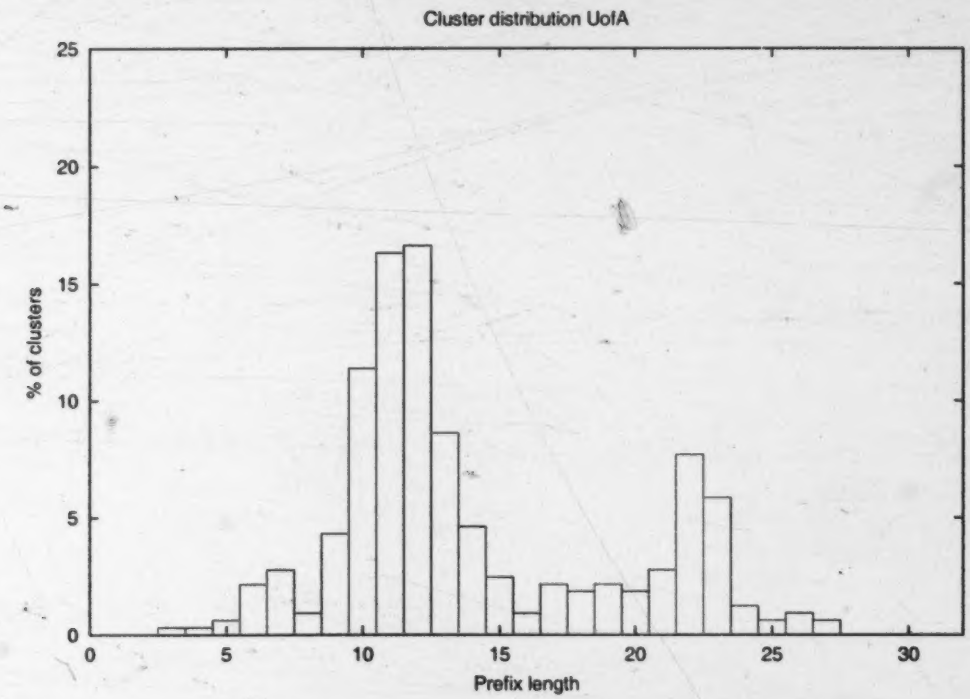


Figure 4.12: Clustering of *UofA* trace.

have prefixes between 10 and 19 bits in length. That is, 86.9% of the clusters are concentrated in 31.3% of the address space. Approximately 95% of all clusters are concentrated in 44% of the whole address space, with prefixes of 9 to 22 bits in length. Similar behaviour is observed for the *SDSC-2* trace (Figure 4.11). This distribution has 88.9% of all clusters with prefix lengths between 12 to 21 bits, which translates to 88.9% of all referenced network regions being concentrated in 31.3% of the address space. Again, approximately 95% of all referenced clusters are located in 44% of the address space with prefixes between 11 and 24 bits in length. It should be noted that for traces *SDSC-1* and *SDSC-2* the regions with high cluster concentration are contiguous. This supports the conjecture that there is the high degree of spatial locality in IP address traces.

The *UofA* trace has a bimodal cluster distribution with two peaks, where 64.3% of all referenced clusters have prefix lengths between 9 and 15 bits and another 16.3%

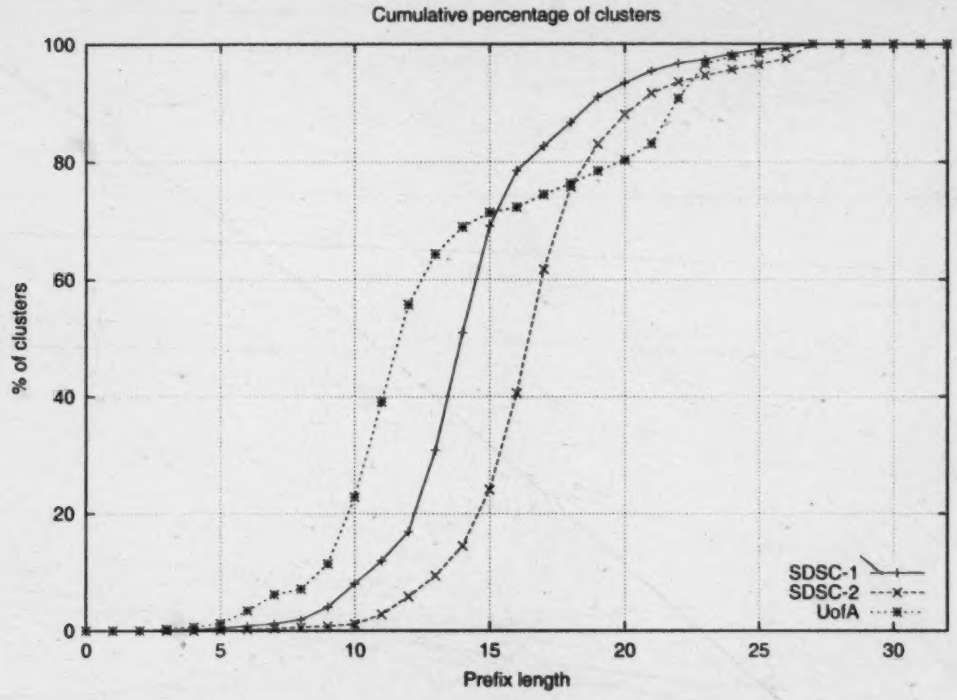


Figure 4.13: Cumulative percentage of clusters. All traces.

of all clusters have prefixes between 21 to 23 bits in length. Thus 80.9% of all address regions are concentrated in 31.3% of the address space. As with previous traces, 90.8% of all clusters have prefix lengths of between 9 and 24 bits, so that 90.8% of all referenced clusters are spread over 50% of the address space.

The clusters or subnets that are referenced in IP traces tend to be in a specific region of the address space. Over 90% of the subnets are concentrated in 50% or less of the whole address space available. This provides additional support to the claim that there is a high degree of spatial locality in IP address traces.

Figure 4.13 depicts cumulative distributions of clusters for all traces. The ranges of prefix lengths that correspond to larger slopes in the graphs are the ranges of high concentration of clusters or subnets. These graphs can be used as a starting point for deriving the parameters that characterize spatial locality in the given traces.

The above data and analysis of spatial locality in IP traffic establish the existence

of a high degree of spatial locality in IP traces and provide the basis for exploiting spatial locality in ways similar of those used in operating systems.

4.4 Summary

Several methods were developed in order to collect the required data and analyze the locality behaviour in IP address traces. For the analysis of spatial locality, a new clustering algorithm was developed that captured the distributions of referenced subnets and enabled an analysis of the degree of spatial locality in traces of references to IP addresses. An analysis of temporal locality was based on the interarrival time distribution of references to the same IP address using well-known sampling techniques.

It was found that a large proportion of references has very short interarrival times in comparison to largest recorded – more than 80% of all IP packets have interarrival times less than 0.250 seconds. It was thus concluded that traces of IP addresses have a very high degree of temporal locality.

Data collected on spatial locality in IP traces showed that over 80% of all subnets referenced are concentrated in approximately 30% of the whole address space and that the references in IP address traces tend to refer to subnets with specific prefix lengths.

The analysis of traces of IP addresses established the existence of both temporal and spatial locality. One of the suggested applications of locality of reference is in using caches to improve system performance. Locality in IP traffic – temporal as well as spatial – can be exploited in ways similar to virtual memory and file systems to improve the performance of IP address lookup systems. That is, the introduction of a locality-aware IP address cache has potential to significantly improve the speed of IP address lookups.

Chapter 5

Multi-zone Cache for IP Address Lookup

The existence of locality in internetwork traffic established in previous chapter can be exploited in a number of ways to improve performance of IP address lookup. One of the prominent and efficient ways of exploiting locality is development of locality-aware caching techniques. When address lookup is performed, IP address cache is searched first and if the IP address is not found in cache, full routing table lookup is performed. Taking into account large amounts of time required to perform routing table lookup, IP address cache is able to speed up significantly IP address lookup. This section describes a novel caching technique which takes temporal and spatial locality into account to accelerate IP address lookup.

5.1 Cache for IP Address Lookup: Design Issues

There are many issues in design of cache memories, and they were described in Chapter 2; however, cache design for IP address lookup is concerned with only a limited number of them. This is because of the nature of cache for network addresses and types of locality that are present in IP traffic. This section describes major aspects of cache design for IP addresses and identifies techniques which are used in designing cache systems that are applicable to the design of IP address cache.

5.1.1 Caching of IP Addresses

In network setting data being cached are different from those that are concern to regular cache memories. Network address cache, in particular IP address cache, is targeted towards caching network layer addresses and their routing information. Routing information can contain port number, interface identifier, or virtual interface identifier where packets with that IP address should be forwarded to. In order for this to work, cache slot or entry (term *cache line* is used for regular cache memories) should have a special format. It should have a search field (or a *tag*) which in this case will be used for storage of IP address, and a key field – used for storage of corresponding routing information. Important design decisions here are the exact cache slot format, number of bits allocated to each field, what information should be stored in each field (i.e. complete IP address or a portion of it).

5.1.2 Related Aspects of Cache Design

Different aspects of cache design were described in Chapter 2 of this thesis. Two of the most important metrics of the performance of cache system are *average access time* and *miss ratio*.

Average access time depends on miss ratio, time required to service a cache miss, and time required to service a cache hit, or a cache access time. To increase the performance of cache, average access time must be minimized. When dealing with the regular cache memories average access time means how much time on average it takes to search both cache and main memory for the requested item. In case of caching IP addresses this translates to average time required to search network address cache in case of cache hit, and average time of routing table lookup in case of a cache miss, which is extremely time consuming process in comparison to regular memory access. Since average access time depends on miss ratio, it makes sense to investigate methods of minimizing miss ratio to reduce average access time.

Miss ratio reflects how often access to cache will result in a cache miss, i.e. the requested information is not found in cache. Minimization of miss ratio is one of the

goals of cache design. As a result, major concern of this research is to design cache system for IP address lookup that has minimal miss ratio under given conditions. One of the ways to minimize miss ratio is to increase cache size (number of cache slots), since when more slots are available in the cache – more items can be stored, thus decreasing the probability of not finding the item in the cache (cache miss).

Cache replacement policy was found to be a very important parameter of cache system. Since network addresses could be viewed as a sequence of references each having its own arrival time similar to regular memory references, the replacement of entries of IP address cache can be organized under some replacement policy developed for regular cache memories. As previous research on locality and replacement policies for network address caches has shown, LRU replacement policy performs very well for cache systems oriented towards caching of network addresses. Thus, LRU is considered as a replacement policy of choice for IP address cache. To assess the performance of network address cache OPT replacement policy can be used as the optimal measure.

Another important characteristic of a cache is a degree of associativity. Due to large number of distinct addresses in the address space governed by Internet Protocol (IP) [Fuller *et al* 93], fully-associative cache is more likely to be used for caching of IP addresses. Fully-associative cache provides the best performance for a given number of cache slots. In addition, it avoids the problem of measurement bias because of specific network addresses in the measurements and makes the results general for any set of network addresses and addressing schemes with similar characteristics [Feldmeier 88]. Current technology enables the development of fully-associative caches at reasonable cost, thus, it is reasonable to select fully-associative mapping for IP address cache. However, if required, different degrees of associativity can be used in implementation of real cache system. In addition, choice of fully-associative scheme simplifies the analytical model of network address cache.

Additional aspect of cache design is the number of cache slots in a cache, or cache size. Smaller caches tend to be faster in searching and updating the information stored; however, larger caches are able to store more items, thus reducing the miss ratio. In this thesis *cache size* is selected as a constraint in design of cache for IP

address lookup. In other words, the design is concerned with how miss ratio can be minimized given a cache of fixed size. Cache size is defined as total amount of storage required (number of bits), not the number of cache slots. This is assumed due to the nature of the caching technique described in this chapter.

Other aspects such as minimizing the delay associated with cache miss and the overheads of maintaining cache consistency are addressed by numerous methods; however, they are not targeted in this research. Cache consistency is a concern because changes in routing table could invalidate forwarding information stored in network address cache.

Based on above discussion the following important characteristics and parameters of IP address cache have been identified: average access time, cache hit time, miss ratio, minimal miss ratio, replacement policy, degree of associativity, cache size, and optimal cache size. IP address cache design should target these aspects in order to achieve high performance metrics.

5.1.3 Performance Analysis of IP Address Cache

To analyze performance of caching system it is necessary to develop some measurement or comparison schemes which enable to identify performance gains. One of the approaches is to build accurate simulation of the cache system and investigate its performance by measuring various parameters such as miss ratio, cache size, and so on. The results obtained using this method are assessed either as absolute values or in relation to results obtained by measuring other caching system with different parameters or organization. In this case, performance analysis is concentrated on investigation of how the parameters such as miss ratio and cache size can be improved with different cache organizations.

As it was suggested in other studies on network address caches, in order to measure the performance of caching system it makes sense to compare the results of simulation to some optimal measure. For example, miss ratio of IP address cache with some replacement policy can be compared to miss ratio of IP cache with LRU and OPT replacement policies. This will provide information on how the choice of

replacement policy affects miss ratio of IP address cache. There are three ways of deriving miss ratio given parameters of a cache. First one is a hardware measurement which is expensive and involves instrumenting an existing system. Second is to develop an analytical model. Analytical models allow to quickly estimate cache performance and provide feedback on how to improve it. Another very popular approach is a trace-driven simulation which suggests evaluating a model of a proposed system using previously recorded address traces. It involves investigation of effects of varying input trace and parameters of cache being modeled on its performance [Agarwal 89].

As a result, a comparison of miss ratios of the cache design in this thesis and regular LRU-managed IP address cache will be a good way of quantifying the performance. Also, the comparison to miss ratio of OPT-governed IP cache can be made in order to relate to the optimal performance that is possible under the given traffic conditions. All comparisons must be done by using the same experimental data such as IP address traces and routing tables.

5.1.4 Miss Ratio Prediction

As it was stated previously, there are a number of ways to derive miss ratio. In this research a trace-driven approach is used to extract the performance measurements from a cache. However, an analytical method which accurately predicts miss ratio of a cache with given parameters would be useful in optimization of the cache design. Some research has been conducted in the area of modeling miss ratio for fully-associative cache memories, it is described in [Singh *et al* 92]. The authors developed a mathematical model of workloads introduced to the system and extracted miss ratio of a finite, fully-associative cache which uses LRU replacement policy under those workloads. Preliminary step in their method was to model the function $u(t, L)$ which they define as the number of unique lines of size L referenced before time t. The observations were made that this function depends on spatial and temporal locality, and the interactions between these two types of locality. The miss ratio was approximated as the time derivative of $u(t, L)$ evaluated at the point when the function has a value equal to the size of the cache (number of entries) [Singh *et al* 92]. As it was

shown, the accuracy of this method of predicting miss ratio is very high for large caches. For small caches this model was close, but not highly precise. Even though this research showed effective way of predicting miss ratio of fully-associative cache it was targeted toward regular cache memories and includes parameters specific to such caches, thus making it hard to adapt to network address caching systems.

Another research presented in [Thiebaut *et al* 92] deals with generation of synthetic traces of addresses for trace-driven simulation of cache memories. Two techniques for generating synthetic traces of addresses with given parameters are described. Major concern in this case was to develop methods that enable generation of traces that possess required parameters such as temporal locality, spatial locality and miss ratio. As this research suggests, miss rate[1] is expressed very accurately in terms of two constants: A and θ, and footprint function [Thiebaut *et al* 92]:

$$u(n) = A * n^{\frac{1}{\theta}} \tag{5.1}$$

where A is related to the working set, θ is a locality characteristic, and $u(n)$ is the number of unique references observed at reference n.

The miss rate for a cache of size C is approximately equal to the slope of the curve of $u(n)$ at the point where $u(n) = C$. In order to find the slope, the above equation (5.1) is solved for n as a function of u, and derivative $\frac{dn}{du}$ is found. The inverse of this derivative is the estimated miss ratio [Thiebaut *et al* 92].

$$n = \left(\frac{u}{A}\right)^{\theta} \tag{5.2}$$

After taking the derivative of (5.2), inverting it, and substituting C for u, miss rate is expressed as follows:

$$\text{Miss Rate} = \left(\frac{dn}{dC}\right)^{-1} = \frac{A^{\theta}}{\theta} * C^{(1-\theta)} \tag{5.3}$$

where C is the size of fully-associative cache expressed in memory words (cache slots) [Thiebaut *et al* 92].

As it was shown in this study, the method of estimating miss rate of a finite fully-associative cache as it is defined in (5.3) is accurate enough to be used in generation

[1] Terms miss rate and miss ratio are used interchangeably in this research

of synthetic traces of addresses for trace-driven simulation of cache memories.

This research showed a method which can be used to predict miss ratio of regular fully-associative cache. There are a number of similarities in referencing behaviour of workloads on regular cache and in a stream of IP addresses. Research presented in [Shi *et al* 01] suggested that footprint function (5.1) developed in [Thiebaut *et al* 92] for memory references can be adopted to reflect IP address reference stream. Thus, it makes sense to investigate how this method of prediction miss ratio [Thiebaut *et al* 92] can be adopted for prediction of miss ratio of multi-zone IP address cache.

5.2 Multi-zone Cache Design

This section presents design of a novel caching technique for caching of IP addresses. First, the organization and operation of a simple IP address cache is explained and some parameters of this cache are clarified. Then the detailed description of multi-zone cache is presented.

5.2.1 Single-zone Cache Organization

This section presents design of high performance IP address cache. Such cache will be very useful in the performance evaluation of multi-zone cache presented in this thesis. As it was discussed previously one of the concerns in IP address cache is the information to be cached, the required amount of storage, and format of each cache slot. Since the size of each IP address under IPv4 addressing scheme has length of 32 bits, the search field of each cache entry should be at least 4 bytes in length (the amount of bits that is required to implement replacement policy is not included in this discussion). The key field should be able to accommodate the largest port (interface) identifier at each interconnection device. The format of each entry of a single-zone cache is shown in Figure 5.2 (a). 10 bits for key field are enough to accommodate $2^{10} = 1024$ port identifiers, which is enough for most contemporary routers. For example, latest version (12.2T) of Cisco's IOS software (OS for Cisco routers) has In-

terface Descriptor Block (IDB) that supports from 300 to 4000 interfaces depending on the platform (platforms with support of 10000 interfaces are still in development). Most common IDB limit is 800 interface identifies per platform.

When a packet is received its IP address is looked up in the cache. If the address is found, the corresponding routing information is retrieved from the cache and passed to the forwarding engine which then forwards the packet accordingly. If the address does not exist in the cache, a complete routing table lookup of this address is requested. When routing table lookup returns the required information, IP address and its routing information are stored in the network address cache.

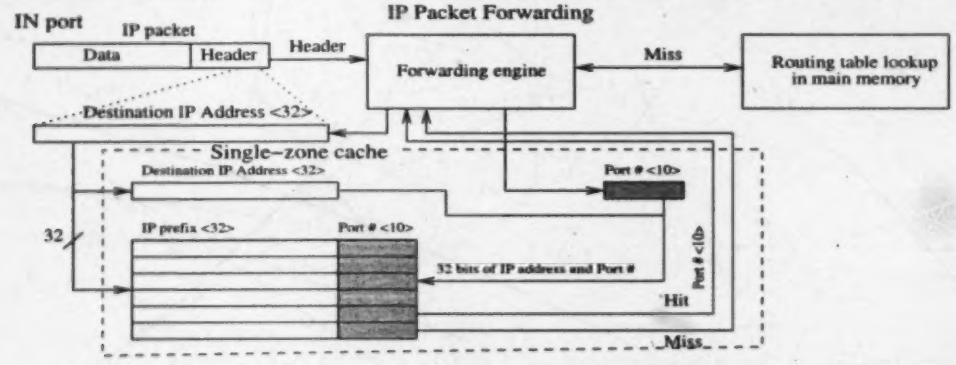


Figure 5.1: Single-zone IP address cache organization.

Taking into account the aspects of IP address cache design presented earlier, a single-zone cache can be designed to have fixed entry format to store the IP address and the routing information; to be fully-associative in order to make the results general and comparable to other studies and to achieve the highest performance for a given cache size; and to be governed by the LRU replacement policy which was found to be very efficient if temporal locality is present in a reference stream. Cache organization described above will ensure high performance for the IP address cache. A diagram of single-zone IP address cache is presented in Figure 5.1. However, such cache design does not take into account the spatial locality present in IP traffic as it is defined in this thesis. Single-zone network address cache stores IP addresses that are located on the same subnet as independent addresses without considering

their relation in space. The technique to exploit spatial locality in regular caches is pre-fetching, but it is not clear how it can be used in network setting: if for IP address to be stored in cache there is no information in routing table on how to reach destinations with *neighbouring* IP addresses then the pre-fetching cannot be performed.

The major goal of this research is to present a novel caching technique that exploits spatial locality and to evaluate its performance. Performance can be evaluated by relating the measurements to the high performance single-zone cache with similar parameters such as cache size, replacement policy and degree of associativity. Based on the above discussion performance evaluation can be done by building accurate simulation of a single-zone cache that allows varying of parameters which are chosen to be the major metrics of performance of caching techniques presented in this research.

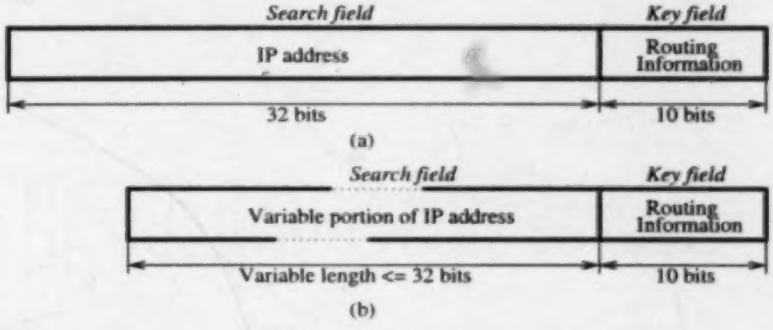


Figure 5.2: Examples of cache entry format.

5.2.2 Multi-zone Cache Organization

Since single-zone IP address cache systems do not take into account spatial locality in IP traffic, cache organizations that exploit spatial locality may outperform caches based on temporal locality only. Such technique is called multi-zone caching of network addresses and is presented in this section.

Caches are usually organized so that they are exposed to all incoming references. However, if most references come from a small portion of the address space, then it

makes sense to allocate most of the cache to the heavily used portion of the address space, and only a few cache entries to the less used portion of the address space. This suggests a cache with multiple areas or *zones*. Based on the spatial locality results presented in the previous chapter, where it was observed that almost 95% of the references come from less than half of the address space, it was conjectured that organizing a cache to also exploit spatial locality would improve performance.

Before presenting the method of partitioning network address cache into multiple zones a few aspects should be clarified. There are two important characteristics of multi-zone cache that make it distinct from simple single-zone cache. First, under multi-zone caching scheme only distinct common prefixes of groups of IP addresses are being cached, not the distinct IP addresses. This reduces the number of potential items to be stored in cache. Second, the restriction on fixed length of search field is removed. In other words, the search field of cache entry can be of variable size as it is presented in Figure 5.2 (b). This reduces overall cache size.

A routing table is a structure that contains a set of subnet addresses (subnet addresses are bundled with their subnet masks) and port identifiers through which the corresponding subnets can be reached. Each subnet address is a portion of IP address that is specified as one of subnetworks in the Internet by the subnet mask. Subnet mask is a 32-bit address mask which is used to indicate which bits of subnet address are being used. Subnet masks are contiguous and represented by single number (mask length) which specifies the number of 'ones' in the mask (Subnet address `192.168.128.0/17` has subnet mask consisting of 17 'ones'). When an IP address is looked up in a routing table the longest prefix match process is used. Subnet mask is applied to the IP address and the resulting quantity (called prefix) is compared to the subnet prefix. For example, an IP address `192.168.219.23` after applying 17-bits mask (`192.168.219.23 AND 255.255.128.0`) becomes `192.168.128.0` which matches a subnet prefix `192.168.128.0/17`. It is possible to find multiple matches in a routing table. If this occurs the match that corresponds to longest matching prefix is selected.

Due to spatial locality in IP traffic a large number of packets whose addresses have common prefix of some length are forwarded toward specific subnet (usually over the

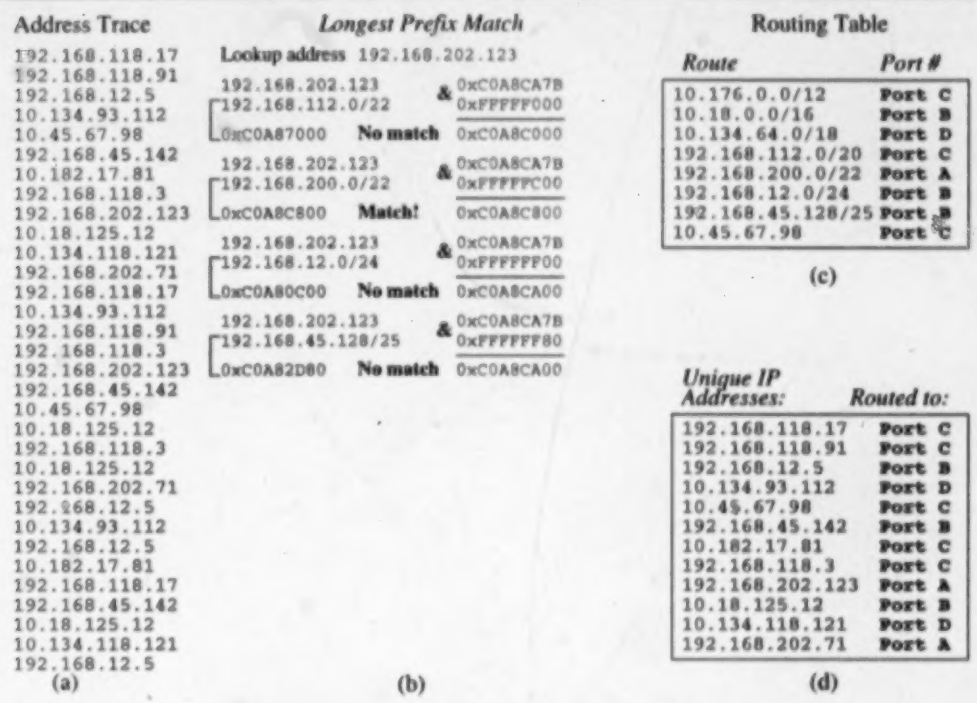


Figure 5.3: IP address trace and routing table.

same interface), so it makes sense instead of caching a number of distinct IP addresses in the cache, to store only the common prefix of the group of addresses and perform search of cache only on this prefix rather than whole IP address. Consider trace of IP addresses in Figure 5.3 (a) and corresponding routing table on the right in Figure 5.3 (c). This trace contains a total of 32 IP addresses with 12 unique IP addresses. The routing table contains 8 entries, each containing IP address of a subnet, its mask, and a port number where the packet with that subnet address should be forwarded. In Figure 5.3 (b) a longest prefix match process for selected unique IP address in the trace is presented. Address, subnet prefixes, and subnet masks are presented in hexadecimal format.

According to the routing table presented in Figure 5.3 (c) packets from the trace in Figure 5.3 (a) will be routed to appropriate ports as it is shown in Figure 5.3 (d). Table 5.1 shows aggregation prefixes for unique IP addresses in the trace. First

column of this table presents IP addresses from the trace in 5.3 (a). Second and third columns specify subnet address (aggregation prefix) and subnet mask length respectively. The last column has port numbers through which the corresponding subnet can be reached. The aggregation of IP addresses is based on common prefix. For example, IP addresses `192.168.118.17` and `192.168.118.91` are aggregated under the common prefix `192.168.112.0/20` because they have matching first 20 bits (`192.168.118.17` = `0xC0A87611` and `192.168.118.91` = `0xC0A8765B`, have common prefix `0xC0A87---`). Detailed description of aggregation and its relation to routing in IP networks is given in [Fuller *et al* 93].

IP Address	Aggregation Prefix	Mask Length	Port #
192.168.118.17 192.168.118.91 192.168.118.3	192.168.112.0	20 bits	Port C
192.168.12.5	192.168.12.0	24 bits	Port B
10.134.93.112 10.134.118.121	10.134.64.0	18 bits	Port D
10.45.67.98	10.45.67.98	32 bits	Port C
192.168.45.142	192.168.45.128	25 bits	Port B
10.182.17.81	10.176.0.0	12 bits	Port C
192.168.202.123 192.168.202.71	192.168.200.0	22 bits	Port A
10.18.125.12	10.18.0.0	16 bits	Port B

Table 5.1: Aggregation of IP addresses for the trace in Figure 5.3 (a).

Aggregation process effectively splits IP address trace into multiple sub-traces each containing groups of addresses aggregated under prefixes that have equal mask lengths. For example, the aggregation described in Table 5.1 splits the trace presented in Figure 5.3 (a) into three sub-traces: one whose addresses are aggregated under prefixes whose mask lengths are in between 0 and 20 bits, a second whose addresses are aggregated under prefixes with mask lengths in between 21 and 24 bits, and the last one whose aggregation prefixes have mask lengths in between 25 and 32 bits. Different sub-traces can be identified depending on prefix lengths selected.

Two different IP address cache configurations are presented in Figure 5.4. The

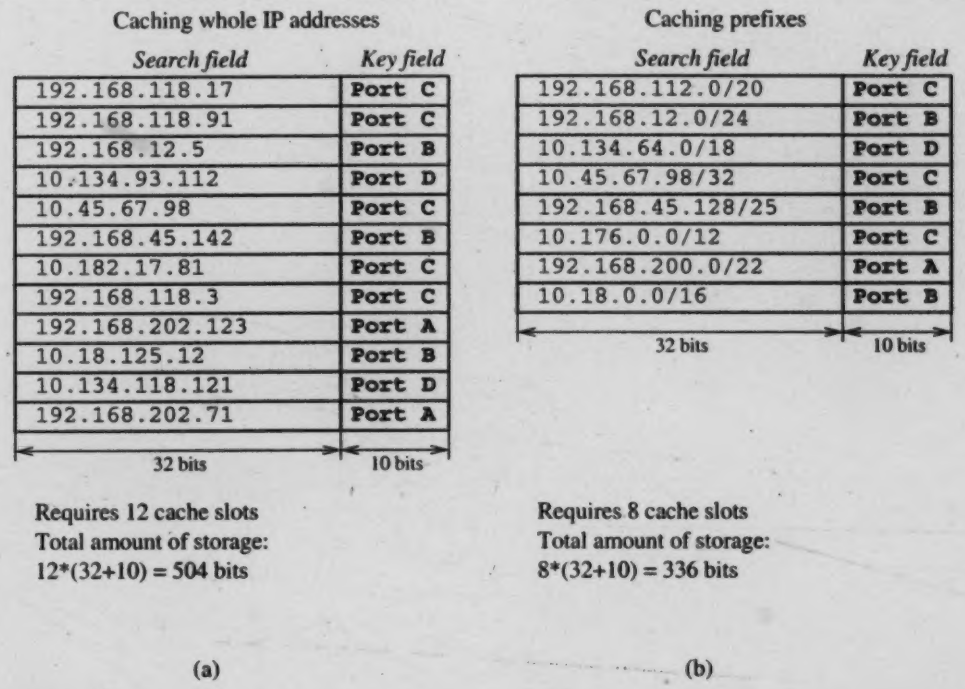


Figure 5.4: Multi-zone cache: caching common prefixes, fixed length search field.

first one, depicted in Figure 5.4 (a), is set up to store complete IP addresses in cache with entry format as shown in Figure 5.2 (a), which is similar to regular IP address cache. The second, presented in Figure 5.4 (b), targeted to cache only common prefixes of groups of addresses that are destined to the same subnet. The common prefix of a group of IP addresses in this case is a subnet address found in the routing table.

Under the caching scheme presented in Figure 5.4 (b), when the IP address is to be stored in the cache, routing table lookup process returns the corresponding port identifier and the subnet address with the mask length in bits. This mask is applied to the IP address and this masked out quantity is stored in cache. When IP address is looked up in such cache, mask of the subnet address is applied to IP address being looked up and this masked out portion is being compared to the information in the search field. If a match is found, the corresponding forwarding information is

retrieved. If not – it is a cache miss and routing table lookup is requested. Since the cache is fully-associative, all entries are searched simultaneously, and the best match selected as a successful search result. One benefit from such caching scheme is that the number of items to be cached is greatly reduced because of aggregation of IP addresses of packets that are destined to the same subnet. Under the conditions described earlier, regular IP address cache presented in Figure 5.4 (a) would require 12 cache slots to achieve miss ratio of 0% (hit rate of 100%) for the given IP address trace, and the cache that caches only common prefixes of groups of IP addresses (Figure 5.4 (b)) would require only 8 cache slots to achieve the same miss ratio. This is due to IPv4 addressing and aggregation scheme [Fuller *et al* 93]. In addition, there is a substantial gain in the efficiency of storage of cached data. Referring to Figure 5.4 regular IP address cache requires 504 bits of storage, where the cache that caches only prefixes consumes only 336 bits. Thus, if the cache size is fixed (in bits), in prefix-only cache organization it is possible to allocate more cache slots than in simple address-only cache organization.

One drawback of such cache organization is that every cache slot has to store or encode in some way the subnet mask of the network address stored in its search field which complicates cache organization and implementation process. As a result, a second technique is targeted to solve this problem and, at the same time, improve performance even further.

Consider the same IP address trace (Figure 5.3 (a)) and the situation presented in Figure 5.5. This time search fields of each cache slot are not required to be of fixed length (Figure 5.5 (b)); furthermore, search fields of cache slots are assigned to store amount of bits that are required to represent only the common prefix (masked out network address) which in many cases is less than 32 bits. So, when caching common prefix `192.168.112.0` only first 20 bits are stored in the search field because the mask length for this prefix is 20 bits. When caching prefix `192.168.200.0` only first 22 leading bits are stored in the search field because the corresponding subnet mask (found in the routing table) is 22 bits inlength. In case of caching the `10.45.67.98` prefix all 32 bits must be stored because in this case the mask covers all 32 bits of the IP address. For prefix the `10.176.0.0` only first 12 bits are cached, since its mask is

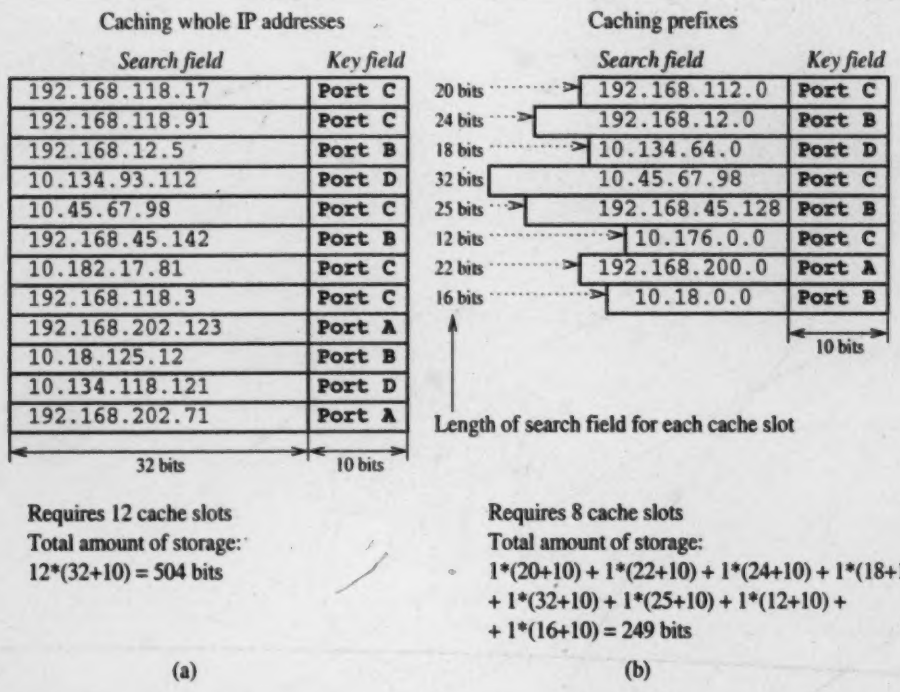


Figure 5.5: Multi-zone cache: caching common prefixes, variable length search field.

12 bits in length. So, instead of caching the whole value `0x0AB00000` (`10.176.0.0`) only value `0x0AB` is cached. Other prefixes are cached in similar way.

When an IP address is looked up in the cache only bits that correspond to the network mask are matched[2]. This way the storage of network masks that correspond to each entry is not required, mask length is implicitly encoded in the search field length, i.e. mask length is equal to length of the search field. The required storage is also reduced. As shown in Figure 5.5, prefix-only cache with variable length search field requires only 249 bits of storage in comparison to 336 bits for fixed width (32 bits) prefix-only cache, and 504 bits for regular IP address-only cache. This is two-fold improvement in storage requirements in comparison to regular IP address cache.

The problem with variable length search fields is that it is very hard to implement

[2]Current technology allows implementation of such searching techniques. Such cache can be implemented using ASIC technology.

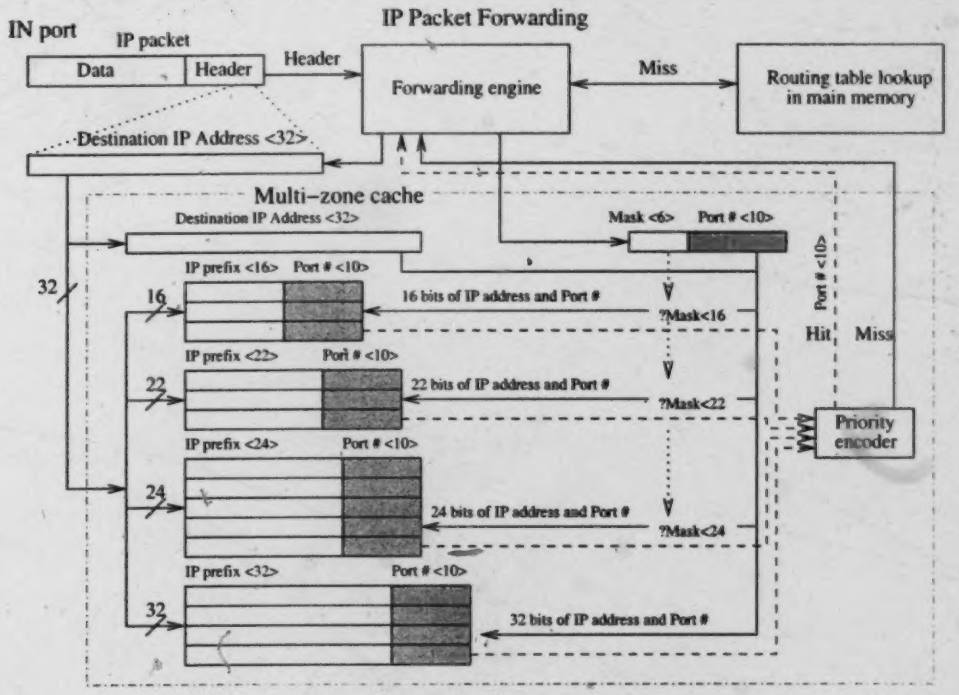


Figure 5.6: Multi-zone IP address cache organization.

in real cache system. Variable length cache slots will cause large overhead for managing the cache that eventually will degrade system performance drawing all benefits of the proposed organization to nil. Thus, generalization of this scheme is required in order to make the described cache organization simpler and implementable.

The length of variable search field is driven by the length of the corresponding prefix, and since there are many prefixes, this method will produce a large number of cache entries with different lengths which is hard to manage and implement. To avoid this complication the notion of *zone* is introduced. Instead of dividing cache into individual entries of variable length, the cache is divided into multiple areas or *zones* where each zone contains a number of cache entries of the same length. In other words, cache slots with the same length are grouped together to form a *zone*. In a multi-zone cache, IP addresses, actually prefixes, are kept in the zone dictated by the local routing table.

An example of multi-zone cache organization is presented in Figure 5.6. This cache has 4 zones with search fields 16, 22, 24, and 32 bits in lengths respectively where each zone is a fully-associative cache. When packet arrives at the input port its header is retrieved by the forwarding engine and destination IP address of that packet is extracted. Then this IP address is looked up simultaneously in all zones of multi-zone cache. The searching in each zone is performed for the exact match of the prefix of destination IP address. The length of this prefix is equal to the width of the search field of the zone. The priority encoder selects the best longest match and returns corresponding port number back to the forwarding engine. If miss occurs, the forwarding engine is notified and complete routing table lookup is requested. When processing of routing information is complete, the forwarding engine determines the appropriate prefix mask length and writes the IP address with forwarding information into multi-zone cache. Zone into which forwarding information should be written is selected based on the mask length, i.e. the zone with the shortest search field which is wide enough to accommodate the number of bits in the mask length is chosen. Then a prefix IP address is written to cache along with the port number. The number of bits in the prefix equals to the width of the search field of the zone selected to store this prefix.

In Figure 5.7 presents three different multi-zone caches with different number of zones. Figure 5.7 (a) shows the example of multi-zone cache configuration that has two zones. First zone has search field of length 18 bits, and second zone has search field of 32 bits in length. Taking into account IP address trace presented in Figure 5.3 (a) and the results of aggregation of network addresses and routing table lookup presented earlier (Table 5.1 and Figure 5.3), packet with destination address `10.18.125.12` is routed toward subnet `10.18.0.0/16` through port `#B`. Since the subnet mask is 16 bits and search field of the first zone is 18 bits, 18-bits portion (`10.18.64.0/18`) of IP address `10.18.125.12` and its routing information, namely port `#B`, are cached in the first zone. This is possible because the IP address portion that carries the prefix information is 16 bits which is shorter in length than 18 bits. The packet with IP address `192.168.202.71` is routed through port `#A` to subnet `192.168.200.0/22`. In this case the subnet mask length is 22 bits which causes the

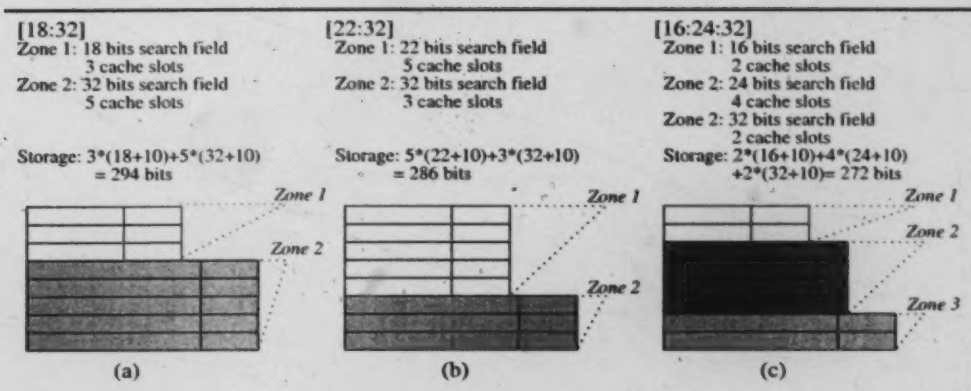


Figure 5.7: Multi-zone cache.

whole IP address `192.168.202.71` to be cached in second zone whose search field allows storage of prefixes between 19 and 32 bits in length. Addresses of packets destined to subnets whose subnet mask length is equal or less than 18 bits are cached in the first zone whose search field is 18 bits in length. Addresses of packets that are to be routed to subnets with masks longer than 18 bits are cached in second zone whose search field is 32 bits in length. Since there are no subnet masks larger than 32 bits in length, it does not make sense to have search field longer than 32 bits.

Similar configuration of multi-zone cache is depicted in Figure 5.7 (b). It also has two zones, but in this case the first zone has a search field capable of storing prefixes of IP addresses of up to 22 bits in length. The second zone is designated to cache prefixes between 23 and 32 bits in length.

A more complicated scenario is depicted in Figure 5.7 (c). There are 3 zones in this cache configuration. The first zone has 16 bit search field, the second zone has 24 bit search field, and the third zone has its search field set to be 32 bits in length. Again, caching of prefixes of IP addresses is distributed among multiple zones. Packet with IP address `10.182.17.81` is routed to subnet with network mask of 12 bits. Since search field of the first zone is 16 bits in length, 16-bits portion (`10.182.0.0/16`) of this IP address is stored in the first zone. Packet whose IP address is `192.168.202.71` is routed toward subnet `192.168.200.0/22` through port `#A`. Since the mask length of this subnet is 22 bits in length and width of search field of the second zone is

24 bits, the 24-bits portion (192.168.202.0/24) of this IP address is cached in the second zone which is wide enough to accommodate 22 bit prefix. On other hand, when packet with address 192.168.45.142 arrives, it is routed through port #B to subnet with network mask of 25 bits, thus, the whole IP address 192.168.45.142 is cached in third zone whose search field is wide enough to accommodate 25 bit prefix. Similarly, when packet with IP address 10.45.67.98 arrives, routing table lookup returns port #C as the next hop for this packet. The mask length is not specified in this case, thus, it is assumed to be 32 bits. The third zone is the only zone that is large enough to accommodate it, since third zone's search field is 32 bits in length.

It should be noted that taking into account the trace of IP addresses and the routing table depicted in Figure 5.3, all caches with configurations presented in Figure 5.7 have the same miss ratio of 0% in a steady state, because they are capable of caching all unique IP addresses present in the trace. For cache in Figure 5.7 (a) appropriate prefixes of 18 bits in length of IP addresses 10.134.93.112 and 10.134.118.121 (aggregated under prefix 10.134.64.0/18 of 18 bits in length), 10.182.17.81, and 10.18.125.12 will be cached in the first zone. Other IP addresses will be put in full into the second zone. For cache with configuration in Figure 5.7 (b) whole IP addresses 10.45.67.98 and 192.168.45.128, which belong to subnets with masks of 32 and 25 bits in length respectively, will be stored in the second zone. Appropriate 22 bits prefixes of the rest of IP addresses will be successfully cached into the first zone. Cache in Figure 5.7 (c) will cache whole IP addresses 10.45.67.98 and 192.168.45.128 in the third zone, 16 bits prefixes of IP addresses 10.182.17.81 and 10.18.125.12 in the first zone, 24 bits prefixes of other IP addresses in the second zone.

Since each zone in the multi-zone caches presented in Figure 5.7 is responsible for caching a group of IP addresses aggregated under prefixes with the same mask length which is equal to or less than the width of a search field of that zone, the trace of IP addresses the caches are exposed to is logically split into different sub-traces. For cache in Figure 5.7 (a) there are two sub-traces. One consists of all addresses which can be aggregated under common prefixes with mask length of 18 or less bits. Other is a group of addresses that are aggregated under the prefixes with mask length be-

tween 19 and 32 bits. There are two sub-traces for the cache configuration presented in Figure 5.7 (b). The first consists of a group of addresses that are aggregated under prefixes with mask length between 0 and 22 bits. Second has addresses whose aggregation prefix's length is in between 23 and 32 bits. For three-zone cache in Figure 5.7 (c), the trace that the whole cache is exposed to is split into three sub-traces: one whose addresses are aggregated under prefixes of length in between 0 and 16 bits, second whose addresses are aggregated under prefixes with mask lengths between 17 and 24 bits, and third that has addresses which are aggregated under prefixes with lengths in between 25 and 32 bits.

The storage requirements for each cache configuration in Figure 5.7 are different. The cache in Figure 5.7 (a) requires in total 294 bits of storage, the cache in Figure 5.7 (b) requires 286 bits of storage, and the cache in Figure 5.7 (c) – 272 bits. This is due to different partitioning schemes performed on the cache to organize it into multiple zones and to select the width of a search field for each zone. If there are more prefixes of specific length to be cached, then more cache slots should be allocated to zones that are capable of caching these prefixes. The distribution of prefixes depends on spatial locality in a reference stream. The optimal multi-zone cache design should take into account spatial locality in IP traffic when allocating space for each zone and selecting width of a search field.

To summarize, a prefix of IP address to be cached should be stored in a zone with shortest search field that is large enough to accommodate it. One zone must be allocated to cache large prefixes of up to 32 bits. All zones in multi-zone cache are fully-associative, so there is no restriction on where in a zone the item should be stored. When cache lookup is performed all zones are searched simultaneously, because they are the parts of a fully-associatove cache, and the best match selected based on longest matching prefix.

In order to design efficient multi-zone cache, the partition of the cache into zones should be based on spatial locality, i.e. allocation of space to each zone should take into account the proportion of references expected to be cached in entries of that zone. The length of a search field for each zone or *zone width* should reflect the spatial locality as well. That is, if there are more references that are destined to subnets

with particular prefix length, this prefix length should be taken into account when choosing zone width. As spatial locality data suggests, a large number of subnets have prefix lengths in a limited sub-set of all possible prefix lengths, depending on the traffic pattern. These subnets are responsible for over 75% of the references. Thus, it makes sense to create zone with width that can accommodate these prefixes.

5.3 Multi-zone Cache Model

To design an efficient cache one should develop a model that accurately describes the cache organization and allows variation of parameters and cache characteristics for the investigation of the cache performance. This section describes a multi-zone cache model that allows the investigation of performance of the cache with various parameters. First, the access time and miss ratio of multi-zone cache are defined, and the ways of optimizing the performance are outlined. Then, a technique to estimate the miss ratio of a multi-zone cache is presented, followed by the description of an optimal multi-zone cache design. Next, validation of multi-zone cache model is discussed.

5.3.1 Access Time and Miss Ratio in Multi-zone Cache

Important performance measures for a regular cache are average access time and miss ratio. These metrics are considered to be appropriate to measure the performance of multi-zone cache with some adaptations.

In typical caches where only one *class* of objects is stored the average access time is:

$$T_{ACCESS} = \mathbf{MR} * T_{MISS} + (1 - \mathbf{MR}) * T_{HIT} \tag{5.4}$$

where **MR** is the miss ratio of the cache, $0 \leq \mathbf{MR} \leq 1$; T_{MISS} is the time required to access main memory when cache miss occurs (miss time), and T_{HIT} is the time required to access cache when cache hit occurs (hit time).

In multi-zone cache there are multiple *classes* of objects that can be cached. IP addresses that are to be cached in different zones are considered to belong to different classes. Different classes of objects have different miss ratios, but identical miss time

(T_{MISS}) and hit time (T_{HIT}) given a fixed cache size. The miss ratios of different classes could be different if, for example, there were many more objects in one class then another, i.e. the distribution of objects between classes is non-uniform. In this case, the class with higher population would have higher miss ratio under a random reference pattern because it is less likely that the object required is in the cache. For IP address cache, class of objects can be a group of references that are destined to the same subnet or a group of subnets, i.e. a group of IP address that have common prefix. If miss and hit times are kept fixed for all classes then the average access time for objects of class j is:

$$t_j = M_j * T_{MISS} + (1 - M_j) * T_{HIT} \tag{5.5}$$

where M_j is the miss ratio of class j.

Then the average access time for multi-zone cache over all references is:

$$\hat{t} = \sum_j f_j t_j \tag{5.6}$$

where f_j is the fraction of references in class j which is defined as:

$$f_j = \frac{n_j}{\mathrm{N}} \tag{5.7}$$

where n_j is the total number of references to objects in class j; N is the total number of references; and with:

$$\sum_j f_j = 1 \tag{5.8}$$

If all the terms in (5.6) were independent, then to minimize average time, each individual term had to be minimized. However, for a cache of fixed size, the M_j's are mutually dependent. As the amount of space allocated to one class is increased to decrease its miss ratio, and so minimize its average access time, the amount of space allocated to other classes must decrease. As a first approximation, assuming a random reference pattern, let miss ratio for zone j be:

$$M_j = 1 - \frac{c_j}{n_j} \tag{5.9}$$

where c_j is the number of objects of class j that are cached, i.e. the number of objects already stored in cache (c_j also can be viewed as the number of entries allocated in

cache for class j, i.e. number of entries in zone j), and n_j is the total number of references to objects of class j. Note that $h_j = \frac{c_j}{n_j}$ is a hit ratio for class j for the cache in a steady state and $\sum_j c_j = \mathbf{C}$ is the cache size assuming that objects of all classes are of the same size.

Assuming that objects of all classes are of the same size, then if there are only two classes:

$$\hat{t} = f_1 t_1 + f_2 t_2 = f_1 * (M_1 * T_{MISS} + (1 - M_1) * T_{HIT}) + f_2 * (M_2 * T_{MISS} + (1 - M_2) * T_{HIT})$$

$$\hat{t} = T_{HIT} + (T_{MISS} - T_{HIT}) * (f_1 M_1 + f_2 M_2)$$

Generalizing back to multiple classes using (5.8):

$$\hat{t} = T_{HIT} + (T_{MISS} - T_{HIT}) * \sum_j f_j M_j \tag{5.10}$$

or equivalently using (5.9):

$$\hat{t} = T_{HIT} + (T_{MISS} - T_{HIT}) * \sum_j \left[f_j \left(1 - \frac{c_j}{n_j} \right) \right] \tag{5.11}$$

In the particular problem of caching of IP addresses, f_j is an attribute of IP traffic and n_j is the distribution of addresses in the network, so that c_j (actually space allocated for the class j) is the only control variable. To minimize access time, the minimum of $\hat{t}$ is defined as:

$$\min(\hat{t}) \equiv \min\left(T_{HIT} + (T_{MISS} - T_{HIT}) * \left(\sum_j f_j M_j \right) \right)$$

since T_{HIT} is fixed and $T_{MISS} > T_{HIT}$ should be true, the above equation becomes:

$$\min(\hat{t}) \equiv \min\left(\sum_j f_j M_j \right) \tag{5.12}$$

or equivalently using (5.9):

$$\min(\hat{t}) \equiv \min\left(\sum_j \left[f_j \left(1 - \frac{c_j}{n_j} \right) \right] \right) \tag{5.13}$$

Going back to an example with only two classes:

$$\min(\hat{t}) \equiv \min(f_1 M_1 + f_2 M_2) = \min\left(f_1 * \left(1 - \frac{c_1}{n_1} \right) + f_2 * \left(1 - \frac{c_2}{n_2} \right) \right)$$

since $\sum_j c_j = \mathbf{C}$ for objects of different classes are of the same size $c_2 = (\mathbf{C} - c_1)$.

$$\min(\hat{t}) \equiv \min\left(f_1 * \left(1 - \frac{c_1}{n_1}\right) + f_2 * \left(1 - \frac{(\mathbf{C} - c_1)}{n_2}\right)\right)$$

$$\min(\hat{t}) \equiv \min\left(f_1 - f_1 * \frac{c_1}{n_1} + f_2 - f_2 * \frac{(\mathbf{C} - c_1)}{n_2}\right) \equiv \min\left(1 - \left(f_1 * \frac{c_1}{n_1} + f_2 * \frac{(\mathbf{C} - c_1)}{n_2}\right)\right)$$

$$\longrightarrow \quad \min(\hat{t}) \equiv \max\left(f_1 * \frac{c_1}{n_1} + f_2 * \frac{(\mathbf{C} - c_1)}{n_2}\right)$$

which occurs at $c_1 = \mathbf{C}$ when $f_1 > f_2$, and at $c_1 = 0$ when $f_1 < f_2$.

This solution suggests that the class that is more frequently referenced should be given more space. Thus, the knowledge that there are multiple classes of objects to be cached can be used to optimize the performance of multi-zone cache. In a real implementation one would not go so far as to completely remove one class from the cache because of temporal locality in the reference pattern, contrary to the assumption made earlier to simplify the analysis, thus making it beneficial to cache IP addresses for recently-seen destinations.

Average access time depends on access times of zones the cache is divided into. In its turn access time of each zone depends on a fraction of all addresses in the IP address trace that are destined to be cached in that zone and a miss ratio for that zone. Since a fraction of IP addresses is a traffic characteristic, then in order to minimize access time of a zone of multi-zone cache miss ratio of that zone should be minimized. This is achieved by allocating more entries to the zone which is responsible for higher proportion of all references going into cache.

According to (5.4) the average access time of a cache depends on a miss ratio of that cache. There are multiple zones in multi-zone cache each having its own miss ratio, thus, the miss ratio of the whole multi-zone cache is called a global miss ratio (**MR**) and it should depend on miss ratios of different zones.

Let m_j be the number of misses occured in zone j, then miss ratio of zone j will be of the form:

$$M_j = \frac{m_j}{n_j} \tag{5.14}$$

Using (5.7) and (5.14):

$$M_j = \frac{m_j}{f_j * \mathbf{N}} \longrightarrow m_j = \mathbf{N} * f_j * M_j \tag{5.15}$$

Global miss ratio for the whole cache is:

$$\mathbf{MR} = \frac{\sum_j m_j}{\mathbf{N}} \tag{5.16}$$

Thus, using (5.16) and (5.15) new expression for global miss ratio of the whole cache becomes:

$$\mathbf{MR} = \frac{\sum_j [\mathbf{N} * f_j * M_j]}{\mathbf{N}} = \mathbf{N} * \frac{\sum_j f_j M_j}{\mathbf{N}}$$

$$\longrightarrow \mathbf{MR} = \sum_j f_j M_j \tag{5.17}$$

where f_j is the fraction of references that are destined to be cached in zone j, and M_j is miss ratio of zone j.

Since f_j is an attribute of a stream of references to IP addresses, the minimization of global miss ratio of a multi-zone cache involves minimization of miss ratios of all zones cache is divided into (similarly to minimization of average access time $\hat{t}$):

$$\min(\mathbf{MR}) \equiv \min\left(\sum_j f_j M_j\right) \tag{5.18}$$

Every zone in multi-zone cache is in itself a simple cache with its own cache size and miss ratio. Thus, the analysis of global miss ratio can be accomplished by analyzing the miss ratio of each zone and combining the results by using (5.17).

According to [Przybylski 90] the miss ratio of a simple single level cache depends on cache size, degree of associativity, and the block size. Block size, as it is defined for regular CPU caches, is non-existent in the context of caching of IP addresses. Since multi-zone cache for IP addresses is chosen to be fully-associative with single set, then each zone of multi-zone cache has the same attributes. Thus, a miss ratio of each zone is a function of size of that zone and is estimated by using methods developed for estimation and prediction of miss ratio of regular cache systems.

5.3.2 Miss Ratio and Cache Size in Multi-zone Cache

The design of high performance multi-zone cache involves minimizing miss ratio of the cache, i.e. global miss ratio. This, in its, turn involves minimizing miss ratio of each zone of multi-zone cache. Different multi-zone cache designs will have different global and per-zone miss ratios. In order to design efficient multi-zone cache with minimal miss ratio different designs have to be evaluated. The accurate miss ratio prediction method enables estimation of global miss ratio and performance evaluation of different multi-zone cache designs.

Previous research suggested that miss ratio of fully-associative cache can be derived using a footprint function (5.1) [Thiebaut *et al* 92]:

$$u(n) = A * n^{\frac{1}{\theta}}$$

where $u(n)$ is the number of unique references observed at reference n, A is a program-specific constant, and θ is a locality characteristic.

As it was shown in [Shi *et al* 01] function (5.1) performs very well in mimicking the behaviour of IP address traces, thus making it attractive in estimating miss ratio of IP address cache. In order to make footprint function (5.1) useful for this purpose, its parameters A and θ must be estimated. Real footprint function was obtained for all traces used in this research by computing the accumulated number of unique references as a function of number of references. Fitting of the proposed footprint function (5.1) was performed by using non-linear regression method available in the `S-Plus` statistical software package [SPlus], [SPlus a], and [SPlus b]. Figures 5.8 through 5.10 show the fitting of footprint function to the number of unique references of three traces used in this research *SDSC-1*, *SDSC-2*, and *UofA*. Table 5.2 shows estimated parameters A and θ for each trace.

From Figures 5.8 through 5.10 we conclude that the footprint function (5.1) can be effectively used in estimating miss ratio of a fully-associative cache because it indeed accurately simulates the behaviour of traces of IP addresses.

The estimated parameters presented in Table 5.2 cannot be directly used in computing miss ratio of a small fully-associative cache due to the following reason. One of the objectives is to keep cache size small, thus, in order to estimate miss ratio as

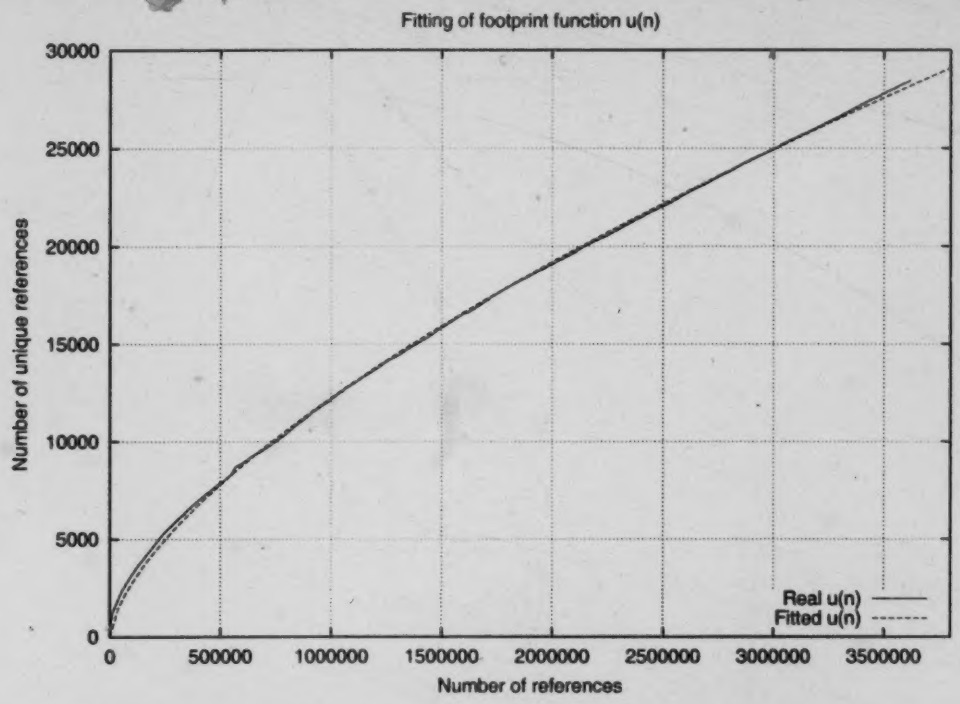


Figure 5.8: Fitting of footprint function $u(n) = A * n^{\frac{1}{\theta}}$. Trace *SDSC-1*.

it is suggested above, the derivative of footprint function (5.1) should be evaluated at the point when $u(n) = c$ (c is the number of cache slots) using small values of c, namely, in order of hundreds. Since values of $u(n)$ for the whole traces used in this research are relatively large (under 6000 for *UofA* trace, 30000 for *SDSC-1*, and over 130000 for *SDSC-2* trace) the fitted footprint function (5.1) with parameters in Table 5.2 may not behave similarly on small values of n. Thus, parameters A and θ of footprint function (5.1) should be estimated using small values of n. Then these

Parameter	SDSC-1	SDSC-2	UofA
A	1.55280	6.296839	8.43141
θ	1.54003	1.734995	2.11689

Table 5.2: Estimated parameters A and θ for all traces.

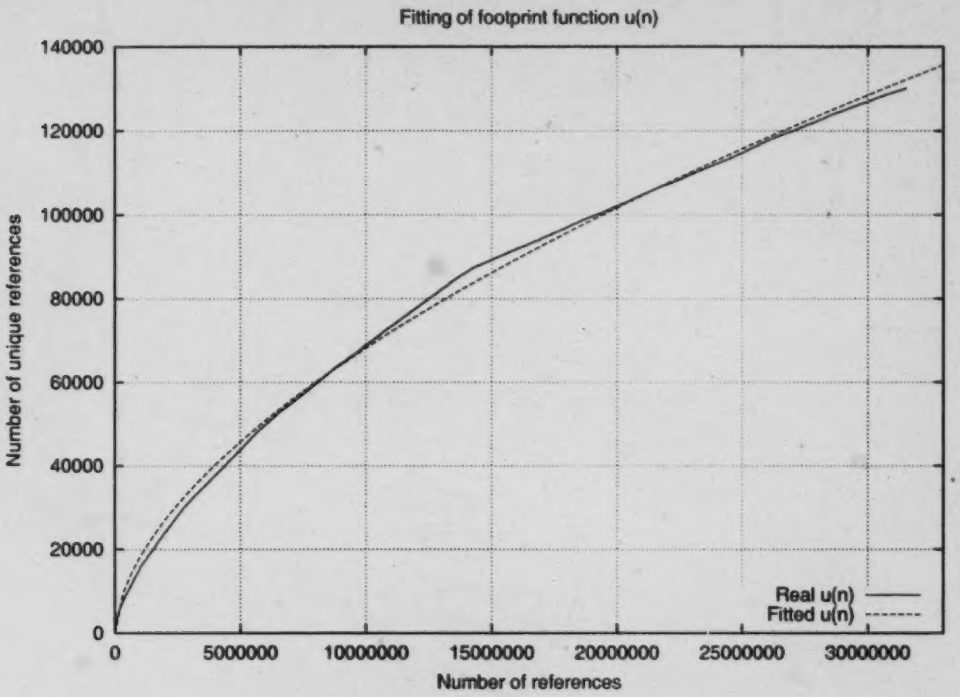


Figure 5.9: Fitting of footprint function $u(n) = A * n^{\frac{1}{\theta}}$. Trace *SDSC-2*.

estimates are used in computing miss ratio.

The fitting of a footprint function (5.1) was repeated with the following modifications. Each trace of references was divided into windows of variable size. The size of each window is the number of addresses referenced which depends on the number of unique addresses referenced in that window. In other words, window is considered to be full when footprint function (5.1) exceeds some predefined value. This value is selected to be the maximum number of slots in the cache because miss ratio of a cache is estimated as the derivative of the footprint function at the point when it equals to the number of cache slots. So, when the footprint function reaches a value that is equal to the number of slots in a cache, its parameters A and θ are estimated and used in computation of miss ratio as defined in (5.3). In these experiments parameters A and θ are estimated on per-window basis, i.e. when the window is filled up the fitting of footprint function is performed and its parameters are obtained. The

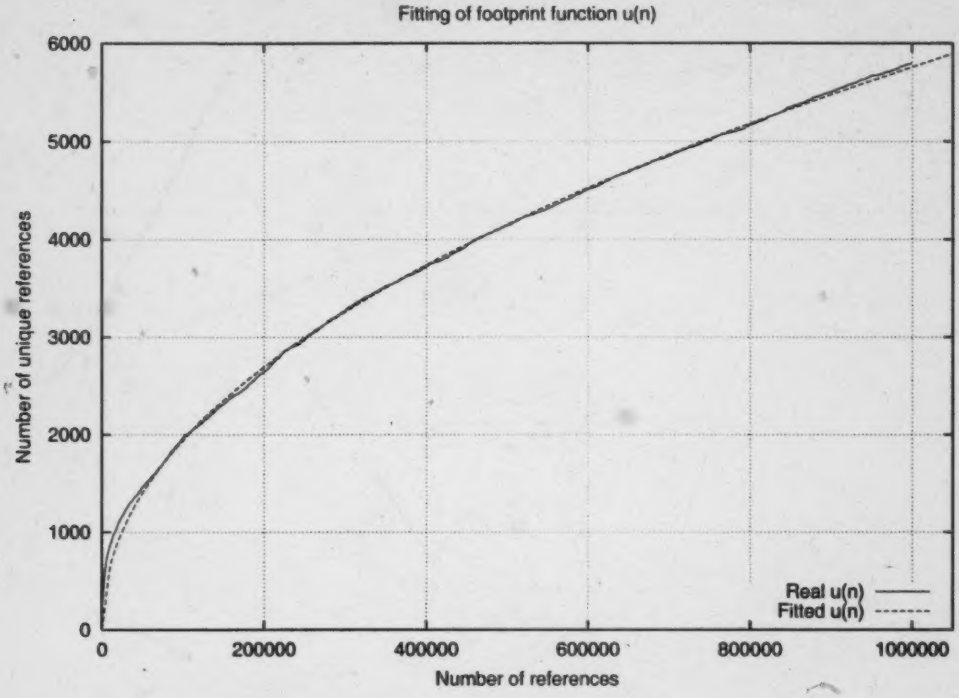


Figure 5.10: Fitting of footprint function $u(n) = A * n^{\frac{1}{\theta}}$. Trace *UofA*.

maximum number of slots in the cache (or in the zone in case of multi-zone cache) in this case was selected to be 400. This is a relatively small number in comparison to other studies, but it allows a multi-zone cache to be implemented as described in this research. Fully-associative caches of small sizes are implementable with current technology.

Figures 5.11 through 5.13 present the results of fitting footprint function (5.1) on per-window basis for all traces as described above. These figures show comparison of fitted footprint function that uses average parameters for all windows and footprint functions of multiple non-overlapping windows that were selected randomly from the window set of each trace. Table 5.3 summarizes values of fitted parameters A and θ for all traces. For the trace *SDSC-1* there were observed 2930 windows for which footprint function reached value of 400. 21569 windows were observed for trace *SDSC-2*, and 623 windows for trace *UofA*. The average window size for each trace is presented in

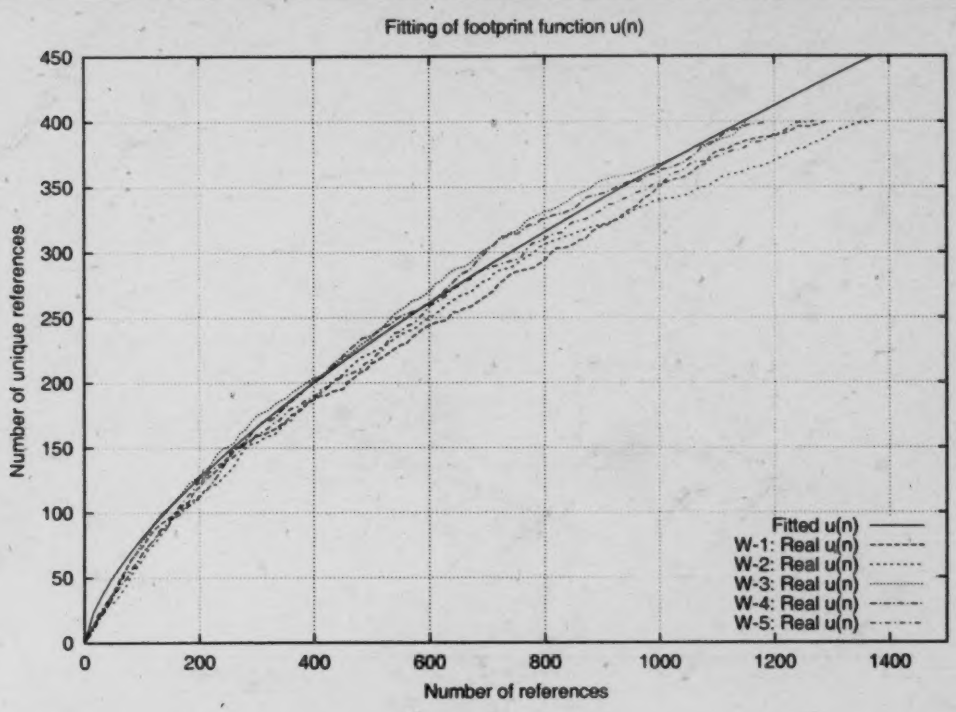


Figure 5.11: Fitting of footprint function $u(n) = A * n^{\frac{1}{\theta}}$ using multiple windows of references. Trace *SDSC-1*.

Table 5.3. The traces with smaller number of windows tend to have lower variation in estimated parameters and average window size.

Trace	Mean A	Std.dev. for A	Mean θ	Std.dev. for θ	Mean Window Size (n)	Std.dev. for Window Size
SDSC-1	3.87538	0.91680	1.51952	0.08956	1235	144
SDSC-2	4.45832	1.22719	1.59976	0.12497	1461	262
UofA	2.43006	0.64472	1.43389	0.08005	1605	112

Table 5.3: Estimated parameters A and θ for all traces; per-window fitting.

From Figures 5.11 through 5.13 it is concluded that footprint function $u(n)$ behaves similarly, only with slight variations, in different windows of the traces. This makes it possible to estimate miss ratio of the cache at any given time using fitted

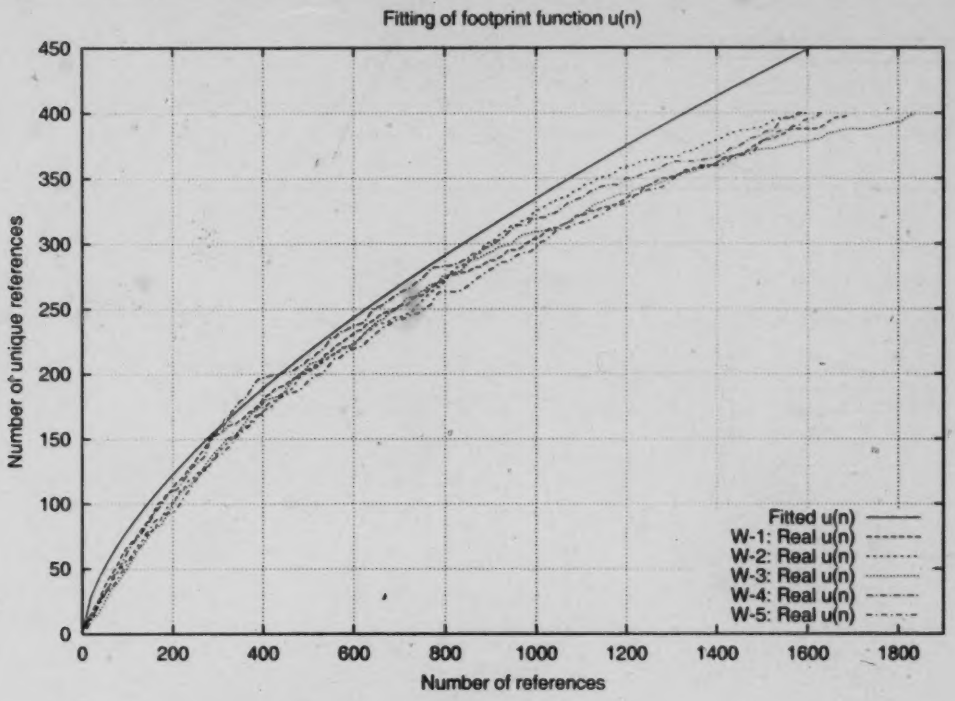


Figure 5.12: Fitting of footprint function $u(n) = A * n^{\frac{1}{\theta}}$ using multiple windows of references. Trace *SDSC-2*.

footprint function (5.1) with parameters presented in Table 5.3. To simplify the design it is possible to obtain accurate estimation of parameters by fitting only the first n references of a trace of IP addresses such as $u(n)$ equals maximum possible number of a cache slots.

According to [Thiebaut *et al* 92] the miss ratio of a fully-associative cache is approximated as a slope of the curve of footprint function $u(n)$ (5.1) evaluated at the point when where $u(n) = c$ (c is the cache size expressed in cache slots). It is defined in (5.3) as follows [Thiebaut *et al* 92]:

$$M = \frac{A^{\theta}}{\theta} * c^{(1-\theta)} \tag{5.19}$$

where M is the miss ratio, c is the number of slots in fully-associative cache, and A and θ are parameters of the footprint function for the trace of IP addresses.

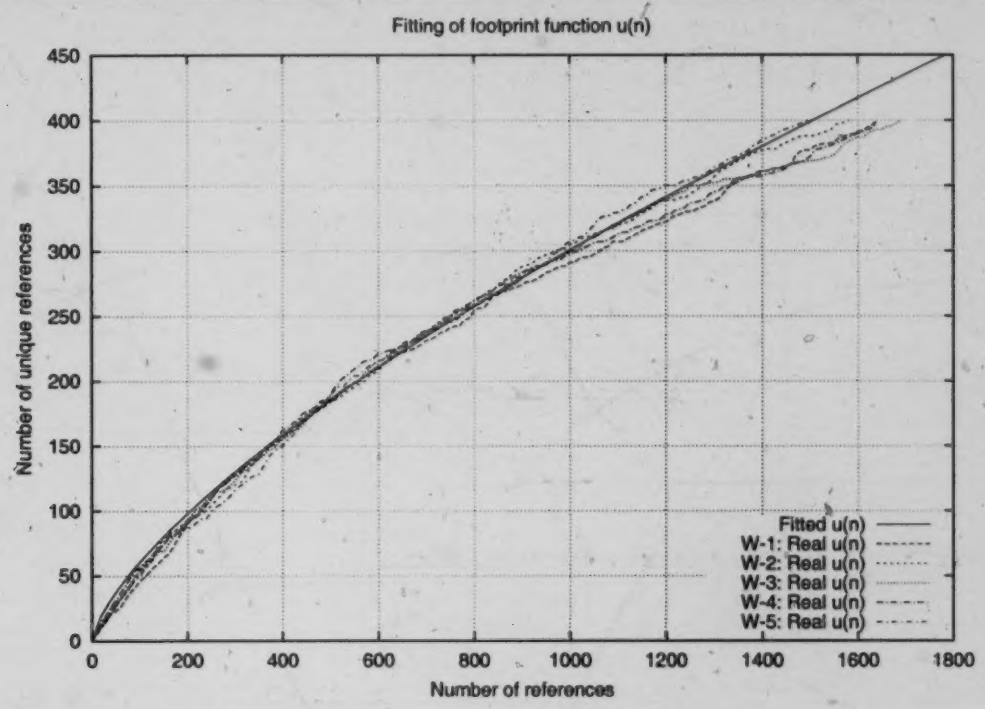


Figure 5.13: Fitting of footprint function $u(n) = A * n^{\frac{1}{\theta}}$ using multiple windows of references. Trace *UofA*.

As it was shown in previous research [Thiebaut *et al* 92] equation (5.19) is very accurate in estimating miss ratio of a fully-associative cache.

Equation (5.19) is a miss ratio for single-zone fully-associative cache with c cache slots. Since every zone of a multi-zone cache is a fully-associative cache, the expression in (5.19) is considered to be a miss ratio of a single zone which has c entries. Every zone is responsible for caching different sets of IP addresses of the whole trace because each zone is configured to cache addresses whose routes have length of their subnet masks equal to or less than the size search field of that zone. This means that the trace of IP addresses to be cached is separated into smaller sub-traces, each sub-trace containing IP addresses that are destined to subnets that have the same length of subnet mask. Thus, each sub-trace destined to be cached in zone j has its own footprint function (5.1) with parameters A_j and θ_j. Taking into account (5.17) and

(5.19) global miss ratio of multi-zone cache becomes:

$$\mathbf{MR} = \sum_j f_j M_j$$

$$\longrightarrow \mathbf{MR} = \sum_j \left(\frac{f_j A_j^{\theta_j}}{\theta_j} c_j^{(1-\theta_j)} \right) \tag{5.20}$$

where f_j is the fraction of references that are to be cached in zone j, c_j is the number of entries in zone j, A_j and θ_j are parameters of a footprint function of references that are to be cached in zone j.

The cache size of a multi-zone cache is expressed in two ways. First, the total number of entries in multi-zone fully-associative cache is defined as:

$$C = \sum_j c_j \tag{5.21}$$

where c_j is the number of entries in zone j.

Second, the total size of multi-zone cache expressed in bits of storage is defined as:

$$\mathbf{C} = \sum_j c_j(w_j + k) \tag{5.22}$$

where c_j is the number of entries in zone j, w_j is the width of the search field of each entry, and k is a constant which represents the length in bits of the key field of each entry in zone.

The total amount of storage available for cache $\mathbf{C}$ is a given parameter defined at the design stage. c_j and w_j are cache parameters that are to be determined during the design stage. The following is the expression for size of multi-zone cache that is to be used in design process:

$$\sum_j c_j(w_j + k) \leq \mathbf{C} \tag{5.23}$$

Expression (5.20) shows that allocating more cache entries to the zone will reduce miss ratio of that zone. According to (5.22) the numbers of cache entries allocated to each zone are mutually dependent, i.e. allocating more cache entries to one zone will reduce the space available for allocation to other zones, thus making miss ratios of

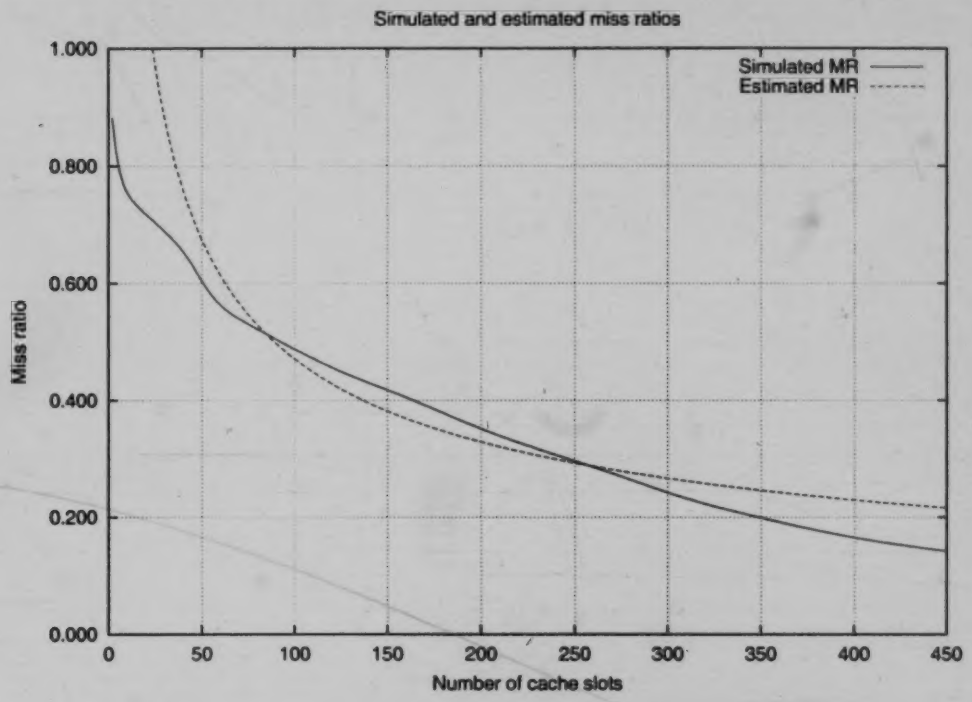


Figure 5.14: Validation of miss ratio estimation technique. Trace *SDSC-1*.

all zones mutually dependent as well because miss ratio of each zone depends on the number of cache entries in that zone. As a result, cache size is a limiting constraint in minimization of global miss ratio of multi-zone cache. Thus, design of multi-zone cache is concerned with minimization of global miss ratio through minimization of sum presented in (5.20) under the given cache size. It should be noted that miss ratio of each zone is a decreasing function of the number of cache entries in that zone.

The optimal multi-zone cache design is to find a set of optimal cache parameters that minimize the global miss ratio. These parameters are selected under the given cache size constraint (5.23) and include a number of entries in each cache zone and width of search field of each zone.

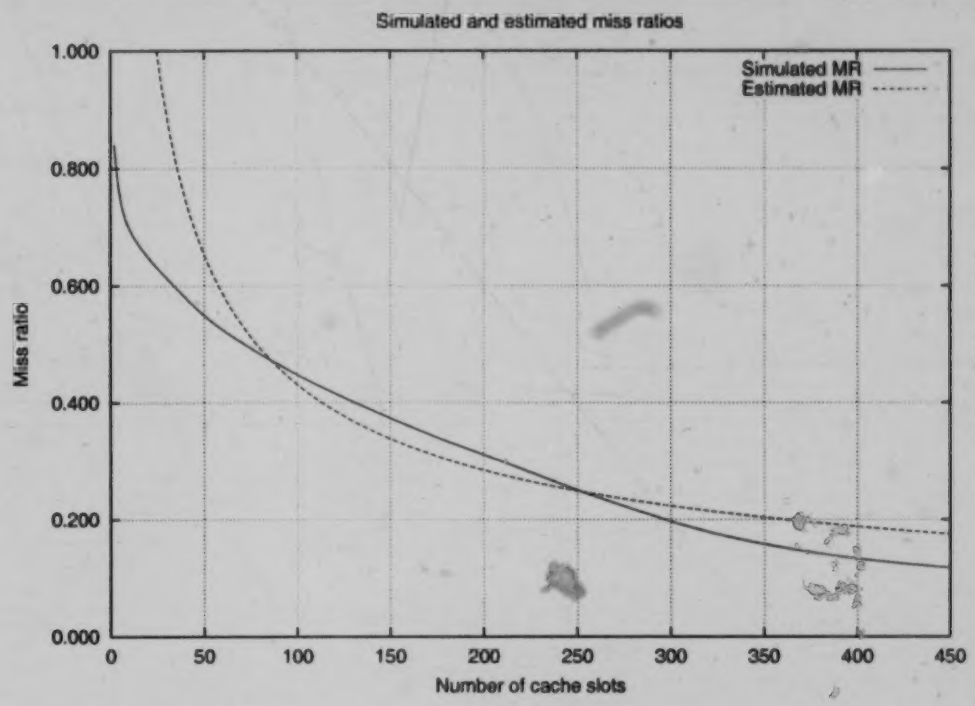


Figure 5.15: Validation of miss ratio estimation technique. Trace *SDSC-2*.

5.3.3 Validation of Multi-zone Cache Model

In order to use the model presented for the design of a multi-zone cache its validity should be assessed, i.e. the accuracy with which it mimics the behaviour of a real cache system should be evaluated. A way to validate the model is to build an accurate simulation of the system and perform the experiments using the same experimental data which were used to derive the parameters of the model. Then the results produced by the simulation are compared to the results produced by the model.

Since each zone of multi-zone cache is a fully-associative cache, it is enough to compare the miss ratio produced by the model for a single zone and the miss ratio produced by the simulation of a single-zone cache. To obtain measurements for a single-zone cache, the simulation tool `SimpleScalar` [SS] was used. This tool is targeted toward simulation of microprocessors and system evaluations. The module

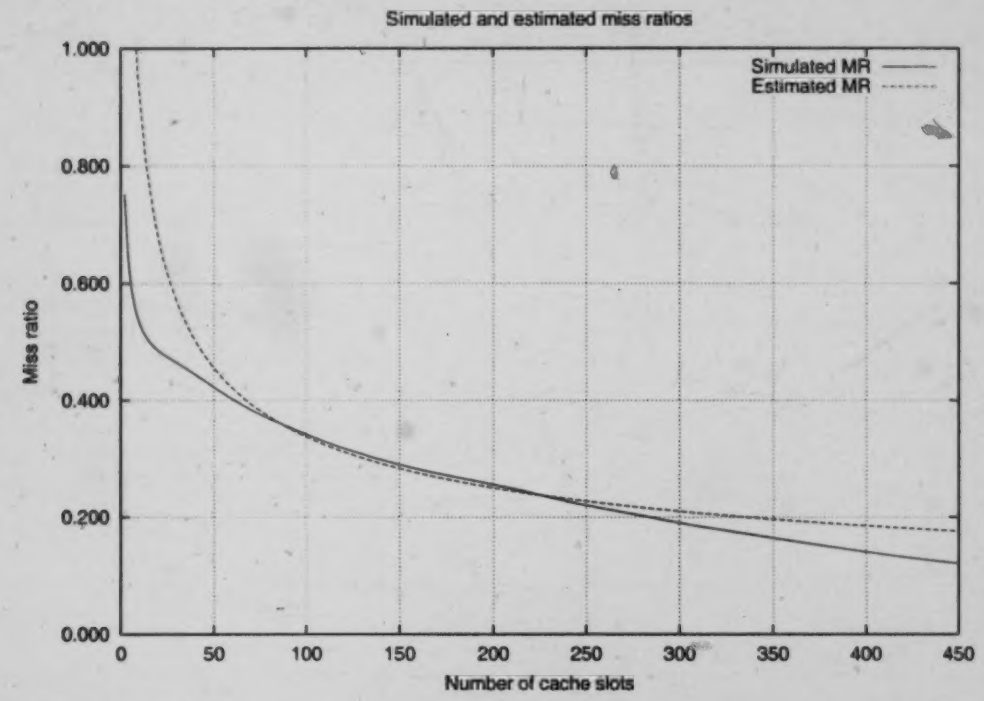


Figure 5.16: Validation of miss ratio estimation technique. Trace *UofA*.

`Cheetah`, which is included in `SimpleScalar`'s tool set, is able to simulate caches with various parameters. It uses a stream of references as an input and implements the required replacement policies, including OPT, and degrees of associativity. Due to the open source nature of these tools, modifications were implemented and incorporated in the `Cheetah` cache simulator to enabled it to handle references to IP addresses and produce miss ratios for caches of different sizes. Since the multi-zone cache model is aimed to help in the design of a fully-associative LRU managed cache, `Cheetah` was setup to simulate such cache with different numbers of cache slots from 2 to 500 with 2 cache slots increments. The input data were the IP address traces used in this research: *SDSC-1*, *SDSC-2*, and *UofA*. The model was evaluated for a single-zone IP address cache with same numbers of cache slots as in simulation. This was repeated for each trace. The values of the parameters A and θ that are used in estimation of the miss ratio in the model (5.19) were taken from Table 5.3 that summarizes the

estimated parameters of the footprint function (5.1) for each trace.

Figures 5.14 through 5.16 present a comparison of the simulated miss ratio produced by the `Cheetah` cache simulator and the miss ratio estimated using the multi-zone cache model. We conclude that the multi-zone cache model performs very well in predicting the miss ratio of these caches. Thus, the multi-zone cache model presented in this thesis can be successfully used to design and evaluate performance of a multi-zone cache.

5.4 Method for Optimal Multi-zone Cache Design

This section describes the method used to design an optimal multi-zone cache based on spatial locality and the multi-zone cache model presented earlier. Based on this model the design of an optimal multi-zone cache with Z zones can be summarized as the optimization problem shown in Figure 5.17. The solution of this optimization problem produces estimates of optimal cache parameters of multi-zone cache. Since the method approximates the miss ratio of a multi-zone cache, these optimal parameters are used as guidelines in the design of an optimal real life cache system.

The optimization problem in 5.17 depends on a number of characteristics that should be identified before-hand.

One of these characteristics is the total amount of storage available for a multi-zone cache $\mathbf{C}$ expressed in bits. This is a platform specific parameter. It is selected based on the resources provided by the platform. These resources include the physical space that is aliowed to be occupied by the multi-zone cache, i.e. how much space is available on the line card; the amount of bits provided by specific hardware for implementation of a fully-associative cache; the backplane type/speed; etc..

Number of zones Z is determined depending on the cache size and the hardware capabilities. In other words, it depends on the overhead for managing multiple zones in terms of number of gates required to implement multi-zone cache, circuitry for priority encoder, and other hardware specific characteristics.

The width of the key field k is selected depending on the required number of

Given input parameters $\mathbf{C}$, Z, k, L, f_j, A_j, and θ_j:

$\mathbf{C}$ is the cache size in bits

Z is the number of zones in a multi-zone cache

k is the width of the key field that holds forwarding information

L is the set of possible widths of the search field of each zone j

f_j is a fraction of references that are destined to be cached in zone j

A_j and θ_j are traffic parameters that are derived by using a footprint function of references that are destined to be cached in zone j,

Find the minimum miss ratio $\mathbf{MR}$ and corresponding sets of optimal cache parameters c_j's and w_j's where $j \in [1, 2, \cdots, Z]$, c_j is the number of cache slots in zone j, w_j is the width of a search field of zone j such that $w_j \in L$ and $w_1 \lneq w_2 \lneq \cdots \lneq w_Z$

$$min\ \mathbf{MR} = \sum_{j=1}^{j=Z} \left(\frac{f_j A_j^{\theta_j}}{\theta_j} c_j^{(1-\theta_j)} \right)$$

subject to:

$$\sum_{j=1}^{j=Z} c_j(w_j + k) \leq \mathbf{C}, \qquad 0 \leq c_j < \left(\frac{\mathbf{C}}{w_j + k} \right), 0 < w_j \leq 32$$

$$j \in [1, 2, \cdots, Z]$$

c_j and w_j are integers

Figure 5.17: Optimization problem.

interfaces that must be supported by the router. The number of required interfaces depends on particular routing platform for which multi-zone cache is being designed.

The set L of widths of the search fields of each zone are selected based on spatial locality. Specific prefix lengths are selected in the ranges of higher concentration of subnets referenced by the IP address traces. These prefix lengths are used as widths of the search fields of multi-zone caches. For example, in case of two-zone cache design, if at least 70% of subnets referenced in the trace of IP addresses have prefix length x, then x should be selected as the width of the search field of the first zone. The search field of the second zone must be selected to be 32 bits in length to cache all IP address which are destined to the rest of subnets with longer prefixes. Other constraints can be used in selection of possible widths of the search field of each zone,

but they are not discussed here.

The knowledge of all characteristics in conjunction with the multi-zone cache design algorithm will enable the designer to evaluate different multi-zone cache designs before costly implementation or prototyping in hardware.

To design a multi-zone cache of total size C with j zones, the width of the search field of each zone is selected based on spatial locality in the reference stream. There must be as zone with search field of 32 bits in length to cache IP addresses that are not cached in other zones with shorter search fields. This is the case when a reference stream has a non-zero fraction of references whose routes have prefix's length of 32 bits. In addition, this requirement allows the multi-zone cache to handle exceptional cases when some IP address must be cached in full, or traffic condition changes in a way that requires caching of prefixes which cannot be cached in other zones. The number of zones in the multi-zone cache is determined based on available system's resources.

The algorithm of optimal multi-zone cache design is presented in Figure 5.18. The complexity of this algorithm is exponential $O(n^Z)$ where Z is the number of zones in the multi-zone cache. Since possible values for prefix length of an IP address are in between 0 and 32 bits, Z will never exceed 32. The algorithm is essentially an exhaustive search over all possible values of the number of cache entries in each zone, thus, it is guaranteed to find the optimal per-zone storage allocation, i.e the number of cache slots in each zone, such that the estimated global miss ratio is minimal. Since the overhead of managing multiple zones in multi-zone cache increases with the number of zones and the cache is constrained by small size and high speed, the number of zones Z will be kept small for most of the designs. As a result, the algorithm presented in Figure 5.18 is considered effective in producing estimation of cache parameters in reasonable time.

To clarify how the algorithm presented in Figure 5.18 can be used to design multi-zone cache consider the following situation. Our objective is to design a two-zone cache with total storage requirements of 12288 bits which is provided by `Xilinx` XCV1000E VIRTEX™ FPGA [Xilinx]. Assuming that this multi-zone cache is responsible for caching of IP addresses presented in trace the *SDSC-1*, the following

$\mathbb{C}$ – size of cache in bits
Z – number of zones
k – width of the key field in bits
L – set of widths of search field
T – trace of IP addresses

optimizeMZ($\mathbb{C}$, Z, k, L, T**)**

1 for all tuples $< w_1, w_2, \cdots, w_Z >$ such that $w_j \in L$ and $w_1 \lneq w_2 \lneq \cdots \lneq w_Z$:
2 find sub-traces t_j that are to be cached in each zone j from trace T $(T = \bigcup_{j=1}^{j=Z} t_j)$
3 for each t_j:
4 compute footprint function of t_j
5 estimate parameters A_j and θ_j by fitting $u(n)$ (5.1) to footprint function of t_j
6 compute fraction of references $f_j = \dfrac{length(t_j)}{length(T)}$
7 find minimum miss ratio of the multi-zone cache:
8 for $\left(c_1 = 1;\ c_1 < \left(\dfrac{\mathbb{C}}{w_1 + k}\right);\ c_1 = c_1 + 1\right)$ do {

$$\text{for } \left(c_2 = 1;\ c_2 < \left(\frac{\mathbb{C} - c_1(w_1 + k)}{w_2 + k}\right);\ c_2 = c_2 + 1\right) \text{ do } \{$$

$$\cdots$$

$$c_Z = \left\lfloor \frac{\mathbb{C} - \sum_j^{j=(Z-1)} c_j(w_j + k)}{w_Z + k} \right\rfloor$$

$$\mathbb{MR} = \sum_j \left(\frac{f_j A_j^{\theta_j}}{\theta_j} c_j^{(1-\theta_j)} \right), \quad j \in [1, 2, \cdots, Z]$$

if ($\mathbb{MR} < \mathbb{MR}_{MIN}$) then {
$\mathbb{MR}_{MIN} \longleftarrow \mathbb{MR}$
$C_{1_{OPT}} \leftarrow c_1, C_{2_{OPT}} \leftarrow c_2, \cdots, C_{Z_{OPT}} \leftarrow c_Z$
$W_{1_{OPT}} \leftarrow w_1, W_{2_{OPT}} \leftarrow w_2, \cdots, W_{Z_{OPT}} \leftarrow w_Z$
}
$\cdots$
}
}

9. return: mininal global miss ratio $\mathbb{MR}_{MIN}$ and sets of corresponding cache parameters:
numbers of cache slots in each zone j: $\{C_{1_{OPT}}, C_{2_{OPT}}, \cdots, C_{Z_{OPT}}\}$
widths of the search field in each zone j: $\{W_{1_{OPT}}, W_{2_{OPT}}, \cdots, W_{Z_{OPT}}\}$

Figure 5.18: Algorithm for optimal multi-zone cache design.

procedure is performed.

The first step is to select width of key field. Assuming that 1024 interfaces will be sufficient, width of the key field is selected to be 10 bits, i.e. $k = 10$ bits.

Next step is to select a set L of possible widths of the search field of the first zone. This selection must be based on spatial locality in the traffic represented by IP address trace (*SDSC-1* in this case). If there is a relatively high concentration of subnets with specific mask lengths that are referenced by the IP address trace, the width of the search field of the zone should be able to accommodate these prefix masks.

Based on spatial locality information presented in Figure 4.13, the set of prefix lengths as potential widths of the search field is selected $L = \{15, 16, 17, 18, 19\}$, because all these prefix lengths correspond to high cumulative proportions of subnets referenced. However, for simplicity, only case where $w_1 = 16$ is presented in this example.

Width of search field of the second zone w_2 is selected to be 32 bits, in order to cache prefixes that will not be cached in the first zone (prefixes whose length is greater than 16 bits). The second zone is responsible for caching of full IP addresses and their routing information.

The algorithm is executed with $\mathbb{C} = 12288$ bits, $Z = 2$ zones, $k = 10$ bits, $w_1 = 16, w_2 = 32$, and with the trace *SDSC-1*.

Sub-traces of references that are destined to each zone are obtained by inspecting the IP address trace and determining the mask length of subnets the IP addresses belong. At the same time, the proportions of references whose subnet masks are 16 bits or less for the first zone, and 17 bits or more (up to 32 bits) for the second zone are found. This is done by accessing the routing table and determining appropriate subnet masks for each IP address in the trace. Then, these sub-traces are used for the estimation of parameters A and θ of the footprint function by collecting first n references such as the $u(n)$ equals the maximum possible number of entries in each cache zone. The maximum number of entries in each zone was selected to be 400 entries because of the maximum amount of available storage. $\mathbb{C} = 12288$ bits available on the targeted hardware which allows maximum of $\frac{12288}{16+10} = 472$ cache entries

in the first zone and zero cache entries in the second zone. Since the second zone cannot be completely eliminated from the cache, the assumed maximum number of cache entries in each zone should be sufficient for the estimation of footprint function parameters in the scope of this design. As a result, the fitting of a footprint function $u(n)$ was performed on first 1795 references ($u(n = 1795) = 401$) of the sub-trace to be cached in first zone and first 4447 references ($u(n = 4447) = 401$) of the sub-trace to be cached in the second zone, because at these values the footprint function equals to the maximum number of cache slots selected. Parameters were found to be $A_1 = 4.75331$, $\theta_1 = 1.64737$ for the first sub-trace and $A_2 = 8.14347$, $\theta_2 = 2.26617$ for the second sub-trace.

Since a routing table was not available for *SDSC-1* trace, clustering data, produced by the clustering algorithm presented in Chapter 4, was used as routing table, where each cluster prefix was used as a subnet address. The fractions were found to be $f_1 = 0.76248$ and $f_2 = 0.23752$.

Then, using the above parameters (A, θ, f_j's, and w_j's) the exhaustive search was performed over all possible combinations of numbers of entries c_j in zone j. It was found that the minimal global miss ratio $\mathbf{MR} = 0.18253$ ($M_1 = 0.18949$ and $M_2 = 0.16019$) and optimal numbers of cache slots for zones are $c_1 = 319$ and $c_2 = 95$. The size constraint is checked by using (5.23):

$$\sum_j c_j(w_j + k) \leq \mathbb{C}$$

$$c_1(w_1 + k) + c_2(w_2 + k) = 319 * (16 + 10) + 95 * (32 + 10) = 12284 \text{ bits } \leq 12288 \text{ bits}$$

Now the optimal parameters (set of c_j's) of the two-zone cache with 16 bits search field for the first zone should be used in simulation of the corresponding two-zone cache in order to obtain a more accurate estimate of its performance.

The same procedure is performed to design a two-zone caches with different widths of the search field of the first zone. The values for the widths of the search field of the first zone are found in the set L. Then the final estimates of all configuration parameters are analyzed and the ones that yield the highest performance are selected.

The presented algorithm is used to design multi-zone cache with an arbitrary number of zones.

The described design method is used to find optimal configuration parameters of a multi-zone cache which in their turn are used to evaluate the performance of different multi-zone cache configurations before actual implementation of a caching system. Since the proposed model takes into account spatial locality present in internetwork traffic, it can be used in design of multi-zone caches that significantly accelerate IP address lookup.

5.5 Summary

To design efficient IP address cache the parameters of such cache and design issues should be carefully investigated. Many aspects of design of regular caches are applied to design of IP address cache with some adjustments. Miss ratio, degree of associativity, replacement policy, and the number of cache slots are identified as the most important aspects of IP address cache design. Global miss ratio is also identified as a measure of performance of a multi-zone cache.

In order to effectively assess the performance of proposed cache organization a comparison to existing high-performance caching schemes is required. Thus, the comparison of miss ratio and cache size of LRU- and OPT-managed caches are selected as major metric of performance.

In addition, effective way to predict miss ratio of multi-zone cache is required to evaluate different cache configurations. It was shown in previous research that there exist a number of ways to predict miss ratio of a regular cache. One of these methods is adopted in this study to accurately predict miss ratio of IP address cache.

This section presented novel locality-based multi-zone caching technique. Major advantage of this technique is exploitation of spatial locality in IP traffic in cache organization. Dividing cache into multiple zones where each zone is responsible for caching IP addresses with certain prefix's lengths and using the aggregation of IP addresses provide substantial gains in cache size which in turn are used to allocate more cache slots, thus reducing miss ratio of a cache.

To identify the performance gains provided by multi-zone cache system a compar-

ison to regular well-designed high-performance cache system is required.

Chapter 6

Performance Analysis of Multi-zone Cache

This chapter presents the analysis of performance gains that are achieved by multi-zone IP address cache which is described in Chapter 5. Multi-zone cache design in addition to temporal locality takes into account spatial locality present in internet-work traffic. Because of high degree of both types of locality multi-zone cache is able to achieve higher performance than regular IP address caching systems.

First, the methodology used in evaluating the performance of multi-zone cache is described. Then, the experiments with different multi-zone cache configurations are presented along with the analysis of the results. In a summary, benefits of using multi-zone caching are presented.

6.1 Methodology

This section describes the methodology of performance evaluation of multi-zone cache. Major performance metric of a multi-zone cache is a global miss ratio. Thus, it makes sense to compare miss ratio of a regular high performance IP address cache to global miss ratio of a multi-zone cache to access performance benefits. In order to do this an accurate model of a regular cache and a multi-zone cache should be created.

6.1.1 Experimental Model for Single-zone Cache

To obtain performance metrics such as miss ratio and cache size of a regular IP address cache a simulation tool `SimpleScalar` [SS] was used, in particular, its cache simulation module `Cheetah`. It was chosen due to its proven record of accurate performance measurement and reach feature set that allows to simulate cache configurations with different replacement policies, cache sizes, and degrees of associativity. The most attractive feature of this simulator was its ability to simulate cache with OPT replacement policy. In case of simulation of IP address cache, miss ratio of a fully-associative cache with LRU and OPT replacement policies is obtained for different cache sizes. As it was noted in Chapter 5, `Cheetah` was modified to handle IP addresses for the purpose of this research.

6.1.2 Experimental Model for Multi-zone Cache

Experimental model for multi-zone cache consists of two parts: analytical and simulation. First, the analytical multi-zone cache model described in Chapter 5 is used to obtain estimates of optimal parameters of a multi-zone cache. Then, these results are used as guidelines for the simulation model of a multi-zone cache. In other words, design of a multi-zone cache is performed using analytical model, then the estimated parameters are used to refine optimal design by performing simulation of a multi-zone cache with optimal parameters.

The analytical model is described in Chapter 5 along with the method for optimal multi-zone cache design.

An accurate simulation model of multi-zone cache was created using `C` programming language that allowed to obtain experimental parameters of a multi-zone cache. Since each zone of multi-zone cache is a fully-associative, LRU-managed cache, then miss ratio of each zone is obtained using simulation and global miss ratio of multi-zone cache is derived using (5.20). Each cache zone is simulated as a stand-alone, fully-associative, LRU-managed cache that receives its sub-trace of IP addresses selected from the whole trace based on the routing table.

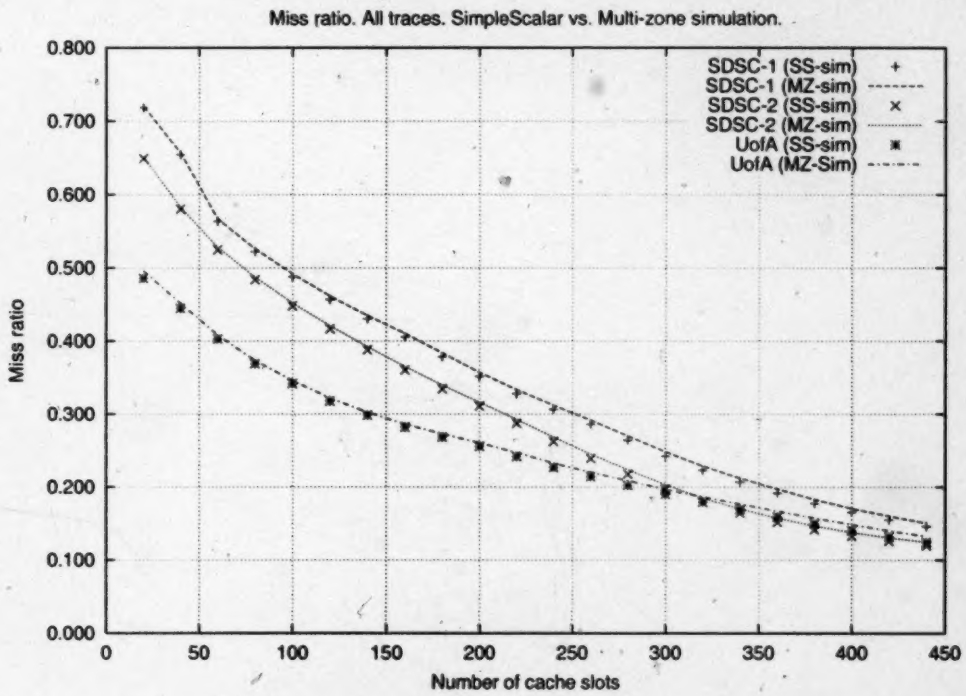


Figure 6.1: Validation of multi-zone cache experimental model.

6.1.3 Experimental Data

IP addresses traces *SDSC-1*, *SDSC-2* and *UofA* are used as experimental data for multi-zone cache simulation. The characteristics of these traces are presented in Table 4.1. Additional information on these traces is required in order to design and simulate multi-zone cache. These are characteristics of sub-traces that are destined to be cached in each zone of a multi-zone cache. Since characteristics of each sub-trace depends on particular cache configurations they are obtained when initial design of multi-zone cache is completed.

6.1.4 Validation of Experimental Models

Multi-zone cache simulation model was implemented using `C` programming language using its standard libraries. Module `Cheetah` of `SimpleScalar` simulation tools package was used to simulate single-zone IP address cache. `SimpleScalar` [SS] is an established tool set for system simulation and evaluation. After a number of experiments it provided no grounds for rejecting it as a simulation tool due to correctness. Thus, it is safe to accept it as an accurate measurement tool which can be used in evaluation of correctness of other simulation models. To check the correctness of multi-zone cache simulator it was configured to simulate one zone with search field of 32 bits in length and was executed with three traces that are used in this research for the same cache sizes as for `Cheetah` simulator. Since the configuration of the multi-zone cache simulated, in fact, corresponds to a single-zone cache it makes it possible to compare its output to the output produced by `Cheetah` module of `SimpleScalar`. The results are presented in Figure 6.1. Cache was not pre-loaded for these experiments. Based on the graph, it is concluded that mutli-zone simulation model is correct and can be used to reliably simulate different multi-zone cache configurations.

6.2 Experiments and Analysis

The experiments were performed using methods described above. Some constraints were set in order to make the results of experiments comparable to findings presented in related research. To compare the performance of a multi-zone cache and a regular single zone IP address cache, the measurements were performed using the same cache size expressed in bits of storage. The size was tied to maximum possible amount of storage allowed by potential hardware which is likely to be used to implement a multi-zone cache. `Xilinx` XCV1000E VIRTEX™ FPGA [Xilinx] was selected as one of the possible devices that can be used to implement a multi-zone caching technique presented in this research. This device allows custom implementation of fully-associative cache with multiple zones and provides a maximum of 12288 bits of storage. The amount that is required to implement replacement policy and other required features

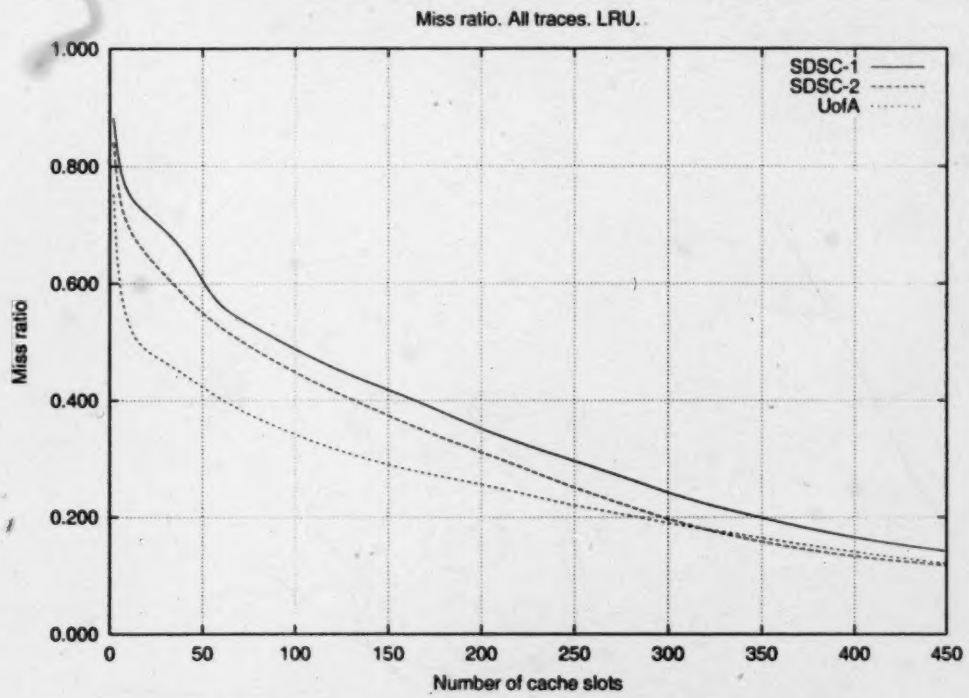


Figure 6.2: Miss ratio of single-zone cache with LRU. All traces.

was not taken into account in order to simplify the performance analysis. Number of cache slots for each cache configuration depends on the format of each entry. Search field for regular single-zone cache is 32 bits in length with key field 10 bits in length which gives maximum of $\lfloor \frac{12288}{(32+10)} \rfloor = 292$ cache slots available for regular IP address cache implementation. In a multi-zone cache every zone has different width of a search field. The key field is 10 bits in length for all zones. This will produce variable number of cache slots depending on the number of zones and size of a search field of each zone.

First, a single-zone regular IP address was simulated for all traces with LRU and OPT replacement policies and with various cache sizes. The results are presented in Figures 6.2 and 6.3. Miss ratios obtained are to be used in analysis of performance of different multi-zone cache configurations. Since there is a cache size constraint, which limits the maximum number of bits of storage for each cache configuration, the

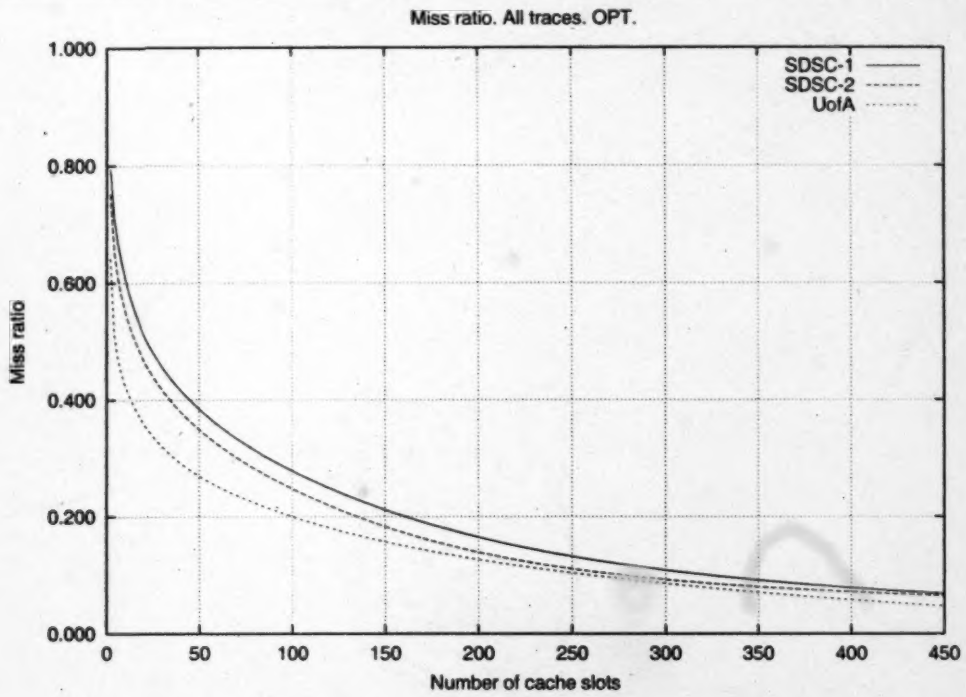


Figure 6.3: Miss ratio of single-zone cache with OPT. All traces.

miss ratios that are used in performance analysis are those that correspond to the maximum size of a regular single zone IP address cache, namely, a cache with 292 cache slots. Miss ratios that correspond to this cache configuration are presented in Table 6.1 and used as minimal miss ratio that a regular single-zone IP address cache is able to achieve for different traces and replacement policies. These values are to be compared to miss ratios of different multi-zone cache configurations.

Replacement policy	Miss ratio for traces		
	SDSC-1	SDSC-2	UofA
LRU	0.25078	0.20502	0.19476
OPT	0.11154	0.09449	0.08813

Table 6.1: Miss ratios for regular single-zone IP address cache with 292 cache slots (12288 bits of storage).

The experiments were designed to evaluate performance of different multi-zone cache configurations and test the method for optimal multi-zone cache design. First step is to perform a design of multi-zone cache and obtain optimal parameters as it was outlined in previous chapter. Then simulation of multi-zone cache with these parameters is performed and the cache configuration that yields highest performance is identified. Finally, the performance of a multi-zone cache is compared to the performance of a regular single-zone IP address cache measured previously. Without loss of generality experiments were conducted for two-zone cache. Design of this cache was based on three IP address traces used in this research: *SDSC-1*, *SDSC-2*, and *UofA*.

Design	Trace	Parameters						
		Search field	Footprint function parameters				Fraction of references	
		(zone 1)	A_1	θ_1	A_2	θ_2	f_1	f_2
S1[14:32]	SDSC-1	14 bits	5.4479	1.7929	12.8188	2.2906	0.4966	0.5034
S1[15:32]		15 bits	6.2734	1.7945	10.2865	2.2929	0.6551	0.3449
S1[16:32]		16 bits	4.7533	1.6474	8.1435	2.2662	0.7625	0.2375
S1[17:32]		17 bits	4.7066	1.6373	6.4929	2.1615	0.7920	0.2080
S1[18:32]		18 bits	8.3349	1.8885	3.3145	1.8794	0.8158	0.1842
S1[19:32]		19 bits	9.0936	1.9034	2.2979	1.8123	0.8928	0.1072
S2[17:32]	SDSC-2	17 bits	4.6889	1.7432	5.6902	1.8717	0.4988	0.5012
S2[18:32]		18 bits	3.4371	1.5800	3.6649	1.7466	0.6264	0.3736
S2[19:32]		19 bits	3.6597	1.5848	4.9071	1.9332	0.7367	0.2633
S2[20:32]		20 bits	3.6105	1.5615	3.4448	1.8107	0.8269	0.1731
S2[21:32]		21 bits	5.7116	1.7325	3.0117	1.8128	0.8746	0.1254
S2[22:32]		22 bits	5.5450	1.7124	2.0353	1.7283	0.9152	0.0848
U[11:32]	UofA	11 bits	16.1587	2.8106	14.9598	2.4222	0.3212	0.6788
U[12:32]		12 bits	17.0204	2.6458	20.9335	2.8518	0.4616	0.5384
U[13:32]		13 bits	15.2219	2.4985	24.9905	3.1292	0.4859	0.5141
U[22:32]		22 bits	7.4501	1.8615	9.7667	3.1049	0.7188	0.2812
U[23:32]		23 bits	6.5266	1.8313	7.3487	2.9499	0.9736	0.0264
U[24:32]		24 bits	6.6512	1.8343	12.9788	3.8882	0.9834	0.0166

Table 6.2: Initial parameters of two-zone cache designs. All traces.

First, prefix lengths are identified that are equal to width of a search field of the first zone. Based on spatial locality data for the traces presented in Table 4.2 and Figures 4.10 through 4.13 specific prefix lengths are selected in the ranges of higher

concentration of clusters that correspond to larger slopes for cumulative cluster distribution. Then, sub-traces and fractions of references that are destined to be cached in each zone are obtained and their footprint function is computed. Next, parameters of a footprint function for each sub-trace are estimated by using `S-Plus` software [SPlus]. All parameters are summarized in Table 6.2 where left most column corresponds to the initial design identification in the form `<TraceID>[<Width Zone1>:<Width Zone2>]`.

There are identified six designs based on each of the traces: *SDSC-1*, *SDSC-2*, and *UofA*. Total cache size was selected to be 12288 bits.

Design	Number of cache slots		Miss ratio		Fraction of references		Estimated Minimal
	zone 1	zone 2	zone 1	zone 2	zone 1	zone 2	Global Miss Ratio
S1[14:32]	190	184	0.1818	0.1798	0.4966	0.5034	0.1808
S1[15:32]	268	133	0.1770	0.1639	0.6551	0.3449	0.1725
S1[16:32]	319	95	0.1895	0.1602	0.7625	0.2375	0.1825
S1[17:32]	326	83	0.1930	0.1557	0.792	0.208	0.1853
S1[18:32]	359	53	0.1559	0.1541	0.8158	0.1842	0.1556
S1[19:32]	383	28	0.1628	0.1664	0.8928	0.1072	0.1632
S2[17:32]	203	162	0.1635	0.1641	0.4988	0.5012	0.1638
S2[18:32]	260	119	0.1769	0.1561	0.6264	0.3736	0.1692
S2[19:32]	286	95	0.1805	0.1598	0.7367	0.2633	0.1751
S2[20:32]	320	64	0.1864	0.1780	0.8269	0.1731	0.1850
S2[21:32]	334	46	0.1674	0.1812	0.8746	0.1254	0.1691
S2[22:32]	347	28	0.1700	0.1745	0.9152	0.0848	0.1704
U[11:32]	161	212	0.0895	0.1422	0.3212	0.6788	0.1253
U[12:32]	211	182	0.1021	0.1338	0.4616	0.5384	0.1192
U[13:32]	211	177	0.1185	0.1236	0.4859	0.5141	0.1211
U[22:32]	296	67	0.1677	0.0546	0.7188	0.2812	0.1359
U[23:32]	338	27	0.1339	0.1969	0.9736	0.0264	0.1356
U[24:32]	323	31	0.1421	0.2700	0.9834	0.0166	0.1442

Table 6.3: Results produced by optimal design algorithm for each configuration.

Initial configuration produced by the optimal design method are used in simulation stage to refine the parameters of two-zone cache. Table 6.3 presents the results produced by the optimal design algorithm given the initial parameters in Table 6.2.

The results presented in Table 6.3 suggest that the best designs for a two-zone cache under the given traffic conditions are the ones that have lowest estimated global miss ratio. To obtain accurate multi-zone cache parameters a simulation of two-zone

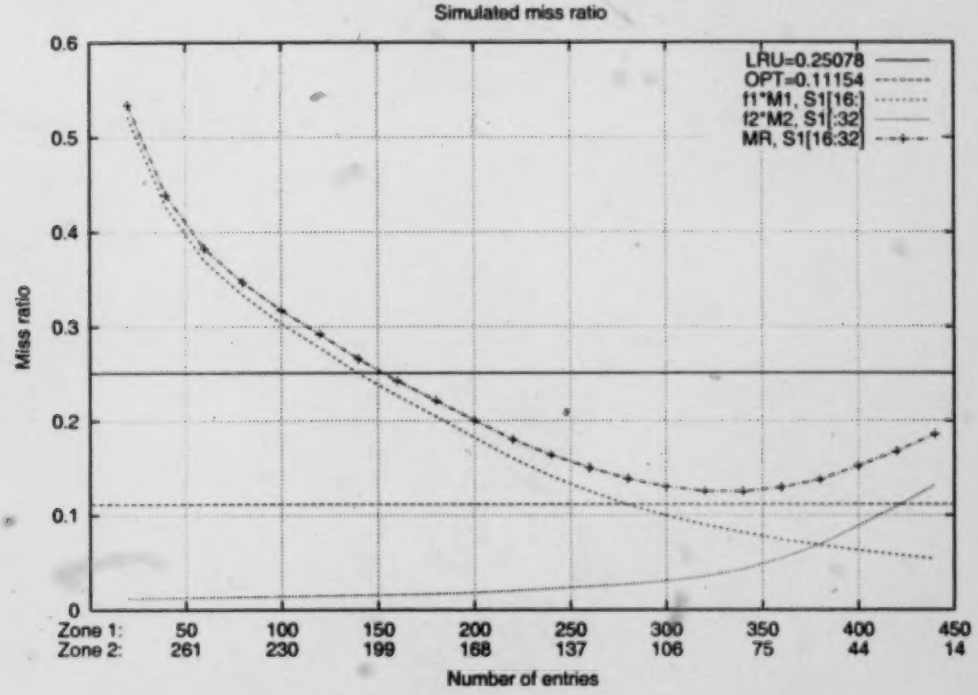


Figure 6.4: Simulated miss ratio of two-zone cache for S1[16:32]

cache is performed for all cache configurations taking into account per-zone space allocation estimates presented in Table 6.3. Estimated number of cache slots per zone for each design provides an approximate per zone storage allocation that corresponds to minimal miss ratio.

Figure 6.4 presents the simulated miss ratio of a two-zone cache configuration that produced the smallest global miss ratio for the traffic presented by the trace *SDSC-1*, namely, S1[16:32]. Figures B.1 through B.5 show simulated miss ratios for the rest of the designs based on *SDSC-1* trace.

Figure 6.5 presents the simulated miss ratio of a two-zone cache design based on *SDSC-2* that produced the lowest global miss ratio – S2[20:32]. Figures B.6 through B.10 show simulated miss ratios for the rest of the designs based on this trace.

Figure 6.6 shows the simulation results for U[23:32] design which is based on *UofA* trace. This design produces the minimal global miss ratio in comparison to

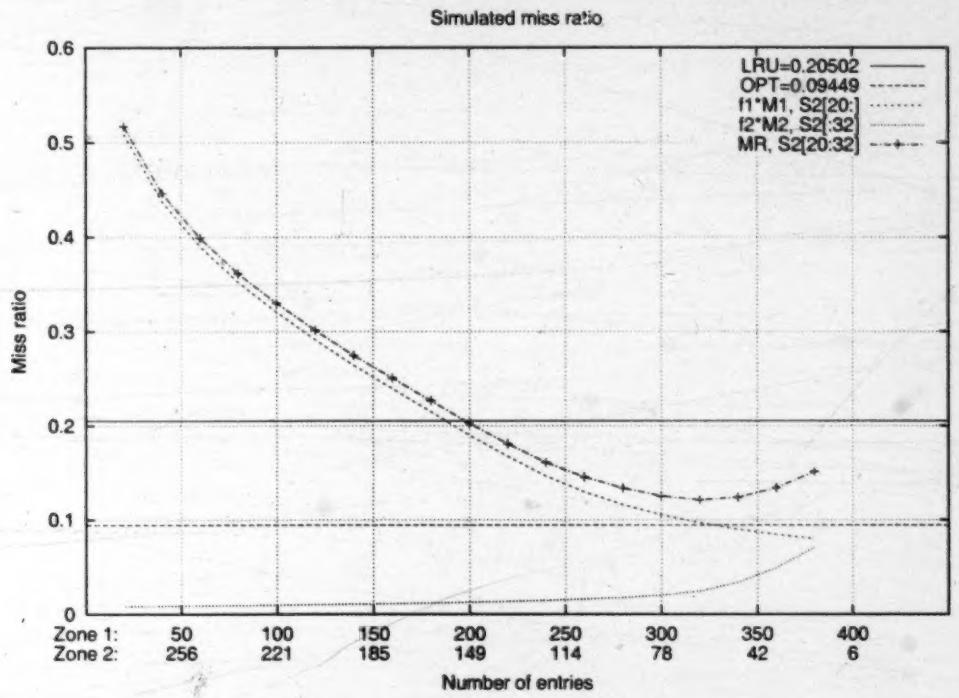


Figure 6.5: Simulated miss ratio of two-zone cache for S2[20:32]

other designs which are based on the same trace presented in Figures B.11 through B.14.

On all figures that show simulated miss ratio of a two-zone cache, miss ratio of high-performance single-zone cache is shown. Two replacement policies are presented – LRU and OPT. As it was stated earlier LRU is implementable in practice, but OPT is impossible to implement. In this case simulated results for OPT replacement policy are show to assess the performance of a two-zone cache.

Minimal global miss ratio achieved by design S1[16:32] is 0.12445. This cache configuration significantly outperforms, in terms of miss ratio, regular IP address cache with LRU replacement policy whose miss ratio was found to be 0.25078 (Table 6.1). In addition, it produces miss ratio that is very close to simulated miss ratio of IP address cache with OPT replacement policy – 0.11154 (Table 6.1).

Optimal multi-zone cache design method produced relatively accurate prediction

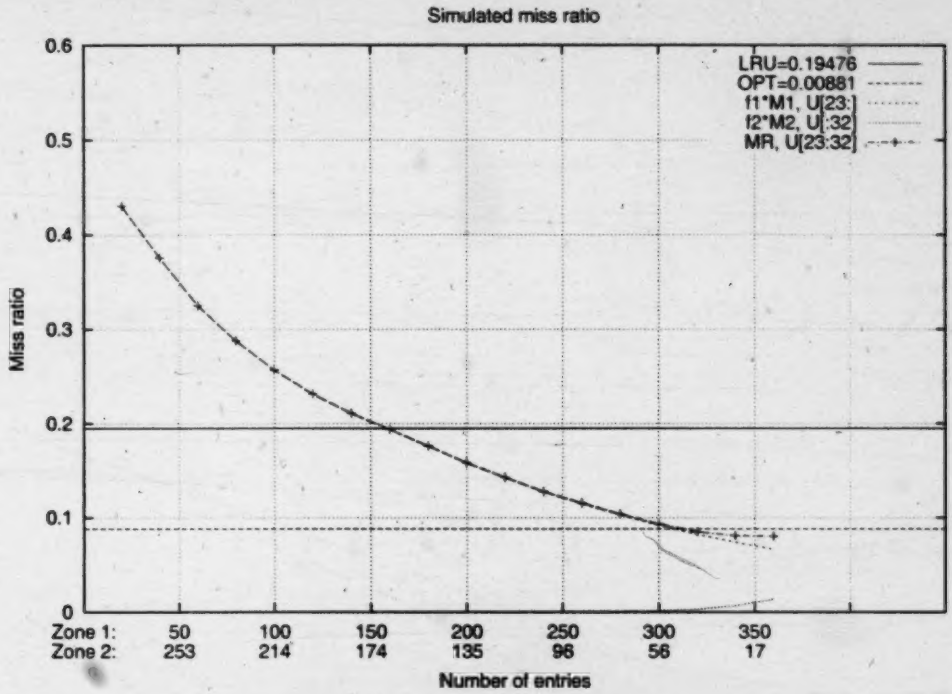


Figure 6.6: Simulated miss ratio of two-zone cache for U[23:32]

of per-zone space allocation for S1[16:32] cache design with the minimal miss ratio. Predicted values were 319 cache slots to be allocated for the first zone and 95 cache slots for the second (Table 6.3). The results for space allocation produced by the simulations were relatively close: 332 cache entries for the first zone and 87 cache entries for the second.

For the designs based on the traffic presented by the trace *SDSC-2* the minimal miss ratio of 0.12097 is achieved by two-zone cache with design S2[20:32]. As with the previous experiment, this cache configuration outperforms regular single-zone IP address cache with LRU replacement policy that achieved miss ratio of only 0.20502. Also, the minimal global miss ratio of two-zone cache with S2[20:32] design is very close to simulated miss ratio of 0.09449 of a single-zone cache with OPT replacement policy.

Per-zone space allocation for S2[20:32] design that produced minimal miss ratio

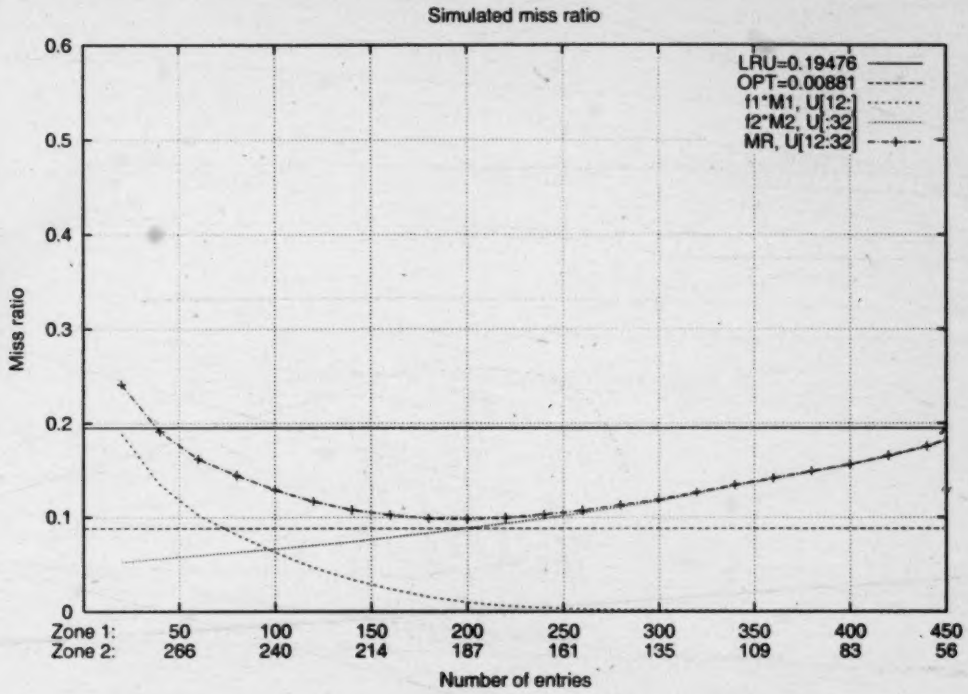


Figure 6.7: Simulated miss ratio of two-zone cache for U[12:32]

was relatively accurately predicted by optimal multi-zone cache design method. There were 320 cache slots predicted for the first zone in comparison to the simulation result of 327 cache slots; and 64 cache slots predicted for the second zone in comparison to 59 cache slots obtained during the simulation.

The minimal global miss ratio for two-zone cache designs based on *UofA* trace is 0.07941 and it is achieved by U[23:32] design. It is more than a two-fold improvement in performance in comparison to a single-zone LRU-governed IP address cache which achieves miss ratio of only 0.19476. In addition, the proposed design outperforms, in terms of miss ratio, a regular cache with OPT replacement policy which produces miss ratio of 0.08813.

The optimal multi-zone cache design predicts per-zone space allocation as 352 cache slots for the first zone (search field width of 23 bits) and 16 cache slots for the second zone (search field width of 32 bits). The simulation produces very close values

for the space allocation scheme for design U[23:32]: 338 cache slots for the first zone and 27 cache slots for the second.

It should be noted that the simulations of designs based on *UofA* trace identified another design, U[12:32], whose performance in comparison to a regular LRU cache is significantly higher. Design U[12:32] produces miss ratio of 0.09792 with per-zone space allocation of 201 cache slots for the first zone and 187 cache slots for the second zone while the corresponding predicted values were 211 and 182 respectively.

The above experiments can be extended to cover different multi-zone cache designs with three or more zones.

From the results presented in Figures 6.4, 6.5, 6.6, and the above discussion it is concluded that two-zone cache significantly outperforms single-zone IP address cache with LRU replacement policy. In some cases, depending on the traffic conditions, two-zone cache is capable of surpassing in performance a regular IP address cache with OPT replacement policy whose performance is considered as non-achievable in practice.

In addition, the optimal multi-zone cache design is considered to be a reliable method for estimating parameters of an optimal multi-zone IP address cache such as the configuration of a multi-zone cache and approximation of its miss ratio.

The comparison of performance of a two-zone cache and regular single-zone IP address cache showed that multi-zone caches have a potential for significantly improving speed of IP address lookup.

6.3 Summary

The design of two-zone IP address cache was performed as it was described in Chapter 5 using IP address traces *SDSC-1*, *SDSC-2*, and *UofA*. Cache configurations with the minimal miss ratios, produced by optimal multi-zone cache design algorithm, were identified. The simulations of a two-zone cache with corresponding parameters were performed and the results were compared to parameters estimated by the optimal design algorithm and to the simulated miss ratios of regular single-zone IP address

cache with LRU and OPT replacement policies.

From the analysis of the results of the simulations it was concluded that a multi-zone cache significantly outperforms a single-zone IP address cache with LRU replacement policy and produces miss ratios very close to those of cache with OPT replacement policy which is considered impossible to achieve in practice. In some cases, multi-zone cache outperformed a regular IP address cache with OPT replacement policy.

The significant performance gain achieved by multi-zone cache over regular single-zone IP address cache provides basis for the claim that multi-zone cache is able to significantly increase speed of IP address lookup.

Chapter 7

Conclusions

Computer networks have revolutionized modern world. They evolved from a handful of interconnected computer systems into a global communication medium that allows groups of people across the globe to share ideas and to collaborate in their work making them more efficient and productive.

Router is a device responsible for the forwarding of traffic across the network. For every packet received by a router, it performs a routing table lookup to determine interface over which the packet will be forwarded toward its ultimate destination. Since routing tables at major routers contain a large number of entries, routing table lookup is a very time consuming process. With increasing speed of networks, routers will have even less time to perform routing table lookup which would result in network congestions, dropped packets, thus reducing performance of the router. In order to improve performance of routing device, IP address lookup must be accelerated.

7.1 Locality in Internetwork Traffic

Previous research in the area of locality in network traffic showed the similarity between referencing behaviour in virtual memory systems and in a stream of network addresses. There were identified two types of locality in network traffic – temporal and spatial. It was suggested that the locality in network traffic could be exploited by introduction of network address caches. However, few of the previous studies have been concerned with locality in large internetworks. One of the goals of this research

was to investigate different types of locality in large IP-based internetworks.

Methods were developed to collect the information needed to analyze temporal and spatial locality behaviour in IP traffic. It was found that a large proportion of references in IP address traces has very short interarrival time which suggested a high degree of temporal locality in network traffic. For analysis of spatial locality a new aggregation-based clustering algorithm was developed that enabled to collect and analyze information on distribution of subnets in IP address traces. It was found that references in IP address traces tend to be to a limited number of subnets concentrated in a small subset of the whole address space. This lead to conclusion that there was a high degree of spatial locality in internetwork traffic.

High degrees of both types of locality in IP traffic make it possible to exploit it to improve the performance of routers and switches through speedup of packet processing, in particular, IP address lookup. This is achieved through the introduction of locality-based multi-zone IP address cache.

7.2 Multi-zone Cache for IP Address Lookup

In computer systems locality in a stream of memory or disk accesses has been exploited through introduction of fast intermediate memory or cache. The examples of such caches include processor caches and disk access caches. These caches are faster than regular memory or a disk system and intended for storing most frequently accessed information. Due to these characteristics caches are able to significantly improve the performance of the whole system.

In the network setting, presence of locality in a reference stream makes it possible to use caching techniques to speedup IP address lookup by storing most frequently referenced IP addresses and corresponding forwarding information in a fast network address cache.

As it was discussed in Chapter 5, there is a number of aspects involved in cache design. Different degrees of associativity, replacement policies and cache sizes are all factors in designing of IP address cache. These issues were included in a multi-

zone cache model developed in this thesis. This model enables a design of efficient IP address cache which takes into account locality present in internetwork traffic by introducing *cache zones*. Each zone of a multi-zone cache is responsible for caching a specific subset of IP addresses. These subsets are determined by prefix length of a subnet to which packets with corresponding IP addresses are forwarded. By allowing cache zones to store only a portion of IP address the total number of cache slots is increased and the number of unique IP addresses that are required to be cached is reduced which results in higher performance of the whole cache.

In addition to the model, a method for optimal multi-zone cache design was presented. It is based on exploiting of both types of locality in IP traffic and enables relatively accurate prediction of performance and optimal configuration of a mutli-zone cache.

The design of two-zone cache and its performance evaluation were performed to show how multi-zone cache model and optimal design method can be applied. Accurate simulation of a high performance IP address cache with LRU and OPT replacement policies was performed. Also, a two-zone cache was simulated. The analysis of the performance of a two-zone cache, which was designed according to the model and optimal method presented, and a high performance regular single-zone IP address cache was performed. The results obtained show that two-zone cache significantly outperforms a regular LRU-based IP address cache and is very close in performance to that of IP address cache with OPT replacement policy which is considered as non-achievable in practice. In some cases, a two-zone cache performed better than IP address cache with OPT replacement policy.

The research presented in this thesis presents a novel multi-zone caching technique along with an optimal multi-zone cache design method. The performance analysis shows that multi-zone cache has strong potential to speedup IP address lookup.

7.3 Directions for Future Research

This research showed one of the ways of how locality in large internetworks can be exploited to improve the performance of network layer devices. It presented a novel multi-zone caching technique along with the optimal multi-zone cache design method which is based on traffic characteristics. This method is relatively accurate in prediction of configuration and performance of an optimal multi-zone cache design; however, the estimation of initial parameters used in this method has potential for improvement. Therefore, one of the possible areas for the future research is to investigate characteristics of internetwork traffic further. These characteristics include measures of temporal and spatial locality.

Another route for extending this research is to develop a prototype of a multi-zone cache using FPGAs and measure its performance. This would enable to evaluate the benefits of introduction of multi-zone cache in real network layer device. In addition, it would help to analyze size/speed/price trade-offs that should be taken into account when designing a multi-zone cache as a part of the larger system. The issues related to presence of multiple caches in one system must be investigated, including cache consistency and cache updates, routing table changes. Also, it would identify hardware specific constraints that could arise at the implementation stage.

Bibliography

[Agarwal 89] A. Agarwal, "Analysis of Cache Performance for Operating Systems and Multiprogramming", Kluwer Academic Publishers, 1989.

[Belady 66] L.A. Belady, "A Study of Replacement Algorithms for Virtual-Storage Computers", IBM Systems Journal, vol.5, no.2, pp.78–101, 1966.

[Bunt *et al* 84] R.B. Bunt, J.M. Murphy, "The Measurement of Locality and the Behavior of Programs", Computer Journal, vol.27, no.3, pp.238–245, 1984.

[Chiueh *et al* 99 a] T. Chiueh, P. Pradhan. "High-Performance IP Routing Table Lookup Using CPU Caching", Proceedings of IEEE INFOCOM 99.

[Chiueh *et al* 99 b] T. Chiueh, P. Pradhan. "Cache Memory Design for Network Processors", Proceedings of IEEE High Performance Computer Architecture Conference (HPCA), 2000.

[Claffy *et al* 93] K.C. Claffy, G.C. Polyzos, H.W. Braun, "Application of Sampling Methodologies to Network Traffic Characterization", Proceedings of ACM SIGCOMM'93, pp. 194–203, 1993.

[Claffy 94] K. Claffy, "Internet Traffic Characterization", PhD Dissertation, Department of Computer Science and Engineering, University of California, San Diego, 1994.

[Degermark *et al* 97] M. Degermark, A. Brodnik, S. Carlson, S. Pink, "Small Forwarding Tables for fast Routing Lookups", Proceedings of ACM SIGCOMM 97, vol.27, no.4, pp.3-14, 1997.

[Denning 70] P.J. Denning, "Virtual Memories", Computing Surveys, vol.2, no.3, pp.153–189, September 1970.

[Deering *et al* 98] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC2460, December 1998.

[Feldmeier 88] D.C. Feldmeier, "Improving Gateway Performance with a Routing Table Cache", Proceedings of IEEE INFOCOM 88, pp.298–307, March 1988.

[Fuller *et al* 93] V. Fuller, T. Li, J. Yu, K. Varadhan, "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", RFC1519, IETF, September 1993.

[Gulati 92] N. Gulati, "Local Area Network Traffic Locality: Characteristics and Application", M.Sc. Thesis, University of Saskatchewan, July 1992.

[Gupta *et al* 98] P. Gupta, S. Lin, N. McKeown, "Routing Lookups in Hardware at Memory Access Speeds", Proceedings of IEEE INFOCOM 98, pp.1240-1247, 1998.

[Hennessy *et al* 90] J.L. Hennessy, D.A. Patterson, "Computer Architecture: A Quantitative Approach", Morgan Kaufmann Publishers, 1990.

[IETF] Internet Engineering Task Force, URL: http://www.ietf.org.

[Jain 90] R. Jain, "Characteristics of Destination Address Locality in Computer Networks: A Comparison of Caching Schemes", Computer Networks and ISDN Systems, vol.18, pp.243–254, May 1990.

[Lampson *et al* 98] B. Lampson, V. Srinivasan, G. Varghese, "IP Lookups Using Multiway and Multicolumm Search", Proceedings of IEEE INFOCOM 98, 1998.

[Maufer 99] T.A. Maufer, "IP Fundamentals", Prentice Hall PTR, 1999.

[McAuley *et al* 93] A.J. McAuley, P. Francis, "Fast Routing Table Lookup Using CAMs", Proceedings of IEEE INFOCOM 93, vol.3, pp.1382-1391, 1993.

[Nilsson *et al* 98] S. Nilsson, G. Karlsson, "Fast Address Lookup for Internet Routers", Proceedings of IFIP 4th International Conference on Broadband Communication, pp.11-22, Stuttgart, Germany, 1998.

[NLANR] NLANR Project, San Diego Supercomputer Center, University of California, San Diego; National Science Foundation Cooperative Agreement No. ANI-9807479; and the National Laboratory for Applied Network Research, URL: moat.nlanr.net.

[Partridge 96] C. Partridge, "Locality and route caches", NFS Workshop on Internet Statistics Measurements and Analysis, February 1996.

[Postel 81] J. Postel. "Internet Protocol", RFC0791, IETF, September 1981.

[Przybylski 90] S.A. Przybylski, "Cache and Memory Hierarchy Design: A Performance-Directed Approach", Morgan Kaufmann Publishers, 1990.

[Rueda *et al* 96] A. Rueda, W. Kinsner, "A Survey of Traffic Characterization Techniques in Telecommunication Networks", Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering 96, vol. II, pp. 830–833, 1996.

[Sklower 91] K.Sklower, "A Tree-Based Packet Routing Table for Berkley Unix", USENIX, pp.93–99, 1991.

[Shi *et al* 01] W. Shi, M.H. MacGregor, P. Gburzynski, "Synthetic Trace Generation for the Internet", The 4th IEEE Workshop on Workload Characterization (WWC-4), Austin, Texas USA, December 2, 2001.

[Singh *et al* 92] J.A. Singh, H.S. Stone, D.F. Thiebaut, "A Model of Workloads and Its Use in Miss-Rate Prediction for Fully Associative Caches", IEEE Transactions on Computers, vol.41, no.7, July, 1992.

[Smith 82] A.J. Smith, "Cache Memories", Computing Surveys, vol.14, no.3, pp.473–530, September 1982.

[SPlus] S-Plus: Statistical Analysis Software Package, MathSoft Inc.

[SPlus a] MathSoft Inc., "S-Plus 6.0 for UNIX User's Guide", Data Analysis Division, Seattle, Washington, October 2000.

[SPlus b] MathSoft Inc., "S-Plus 6.0 for UNIX Programmer's Guide", Data Analysis Division, Seattle, Washington, October 2000.

[SS] SimpleScalar: Simulation Tools for Microprocessor and System Evaluation, URL: www.simplescalar.org.

[Talbot *et al* 99] B. Talbot, T. Sherwood, B. Lin, "IP Caching for Terabit Speed Routers", Proceedings of IEEE GLOBECOM 99, vol.2, pp.1565–1570, 1999.

[Tanenbaum 96] A. S. Tanenbaum, "Computer Networks", Third Edition, Prentice Hall PTR, 1996.

[Thiebaut *et al* 92] D. Thiebaut, J.L. Wolf, H.S. Stone, "Synthetic Traces for Trace-Driven Simulation of Cache Memories", IEEE Transactions on Computers, vol.41, no.4, April, 1992.

[UA] Communication Networks Research Group, Department of Computing Science and Computing and Network Services, University of Alberta, URL: www.cs.ualberta.ca/~networks and URL: www.ualberta.ca/CNS.

[Varghese *et al* 98] G. Varghese, V. Srinivasan, "Faster IP Lookups Using Controlled Prefix Expansion", Computer Communication Review, ACM SIGCOMM 98.

[WAND] WAND Research Group, (Waikato Applied Network Dynamics Research Group), Computer Science Department, University of Waikato, URL: wand.cs.waikato.ac.nz.

[Waldvogel *et al* 97] M. Waldvogel, G. Varghese, J. Turner, B. Plattner, "Scalable High Speed IP Routing Lookups", Proceedings of ACM SIGCOMM 97, vol.27, no.4, pp.25-36, 1997.

[Xilinx] Xilinx Inc. XCV1000E Virtex™ FPGA, URL: www.xilinx.com.

[Xu *et al* 00] J. Xu, M. Singhal, J. Degroat, "A Novel Cache Architecture to Support Layer-Four Packet Classification at Memory Access Speeds", Proceedings of IEEE INFOCOM 2000.

Appendix A

Data Collected on Traces

This appendix contains data that are discussed, but not presented in the main text of this thesis.

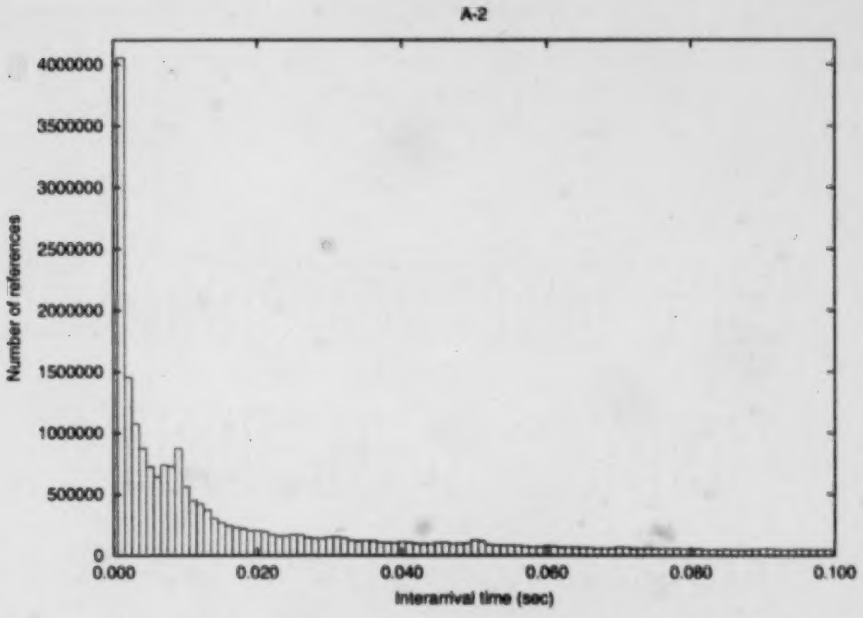


Figure A.1: Number of references with interarrival times from 0.001 to 0.100 sec. Trace *A-2*.

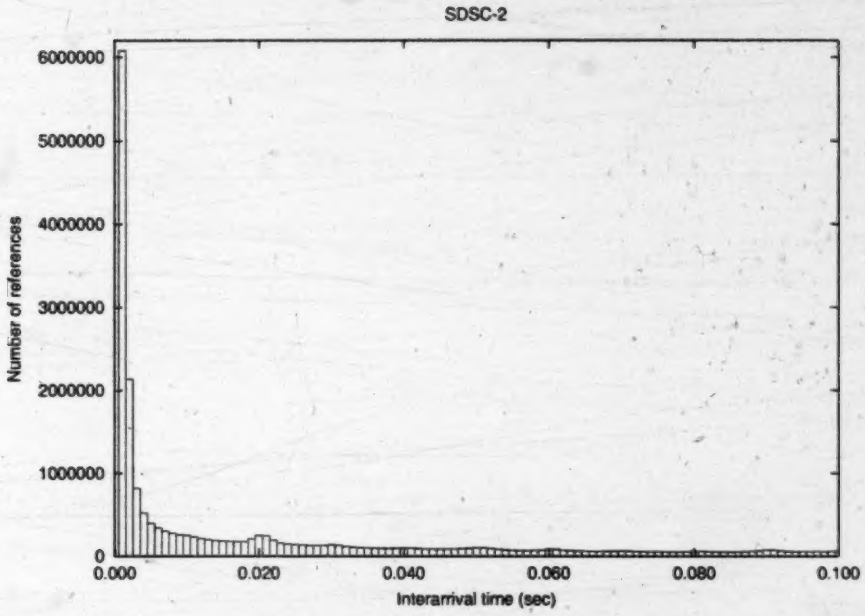


Figure A.2: Number of references with interarrival times from 0.001 to 0.100 sec. Trace *SDSC-2*.

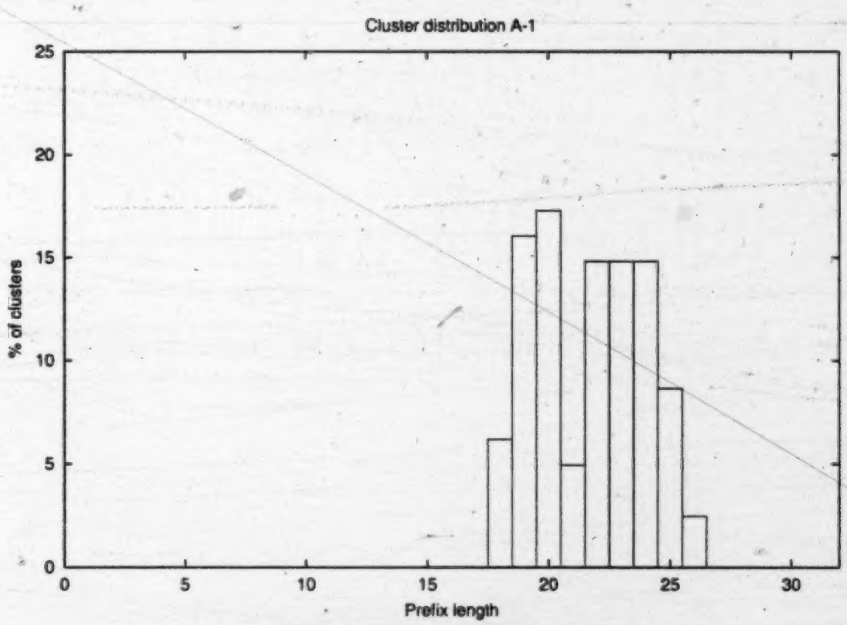


Figure A.3: Clustering of *A-1* trace.

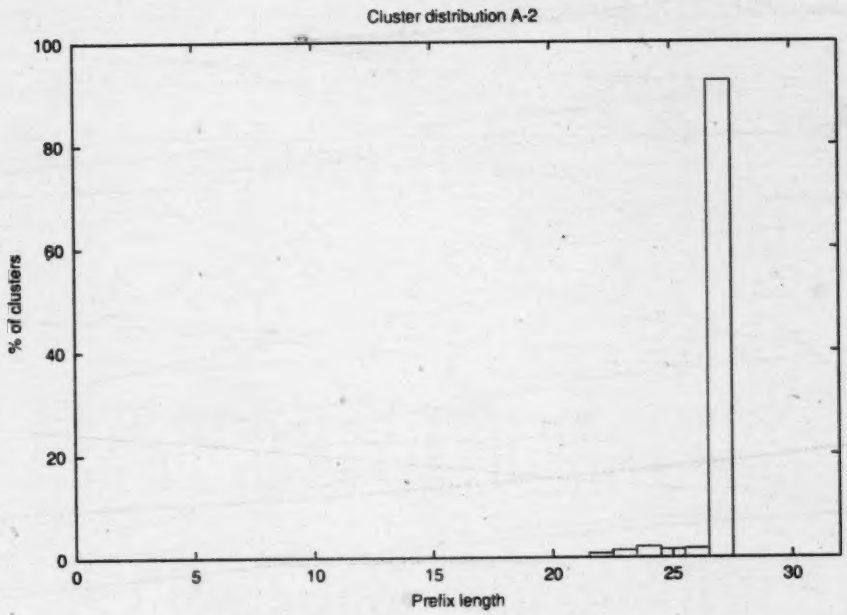


Figure A.4: Clustering of *A-2* trace.

Appendix B

Performance of Multi-zone Cache: Simulation Results

This appendix contains simulation results for different two-zone cache configurations presented in Chapter 6.

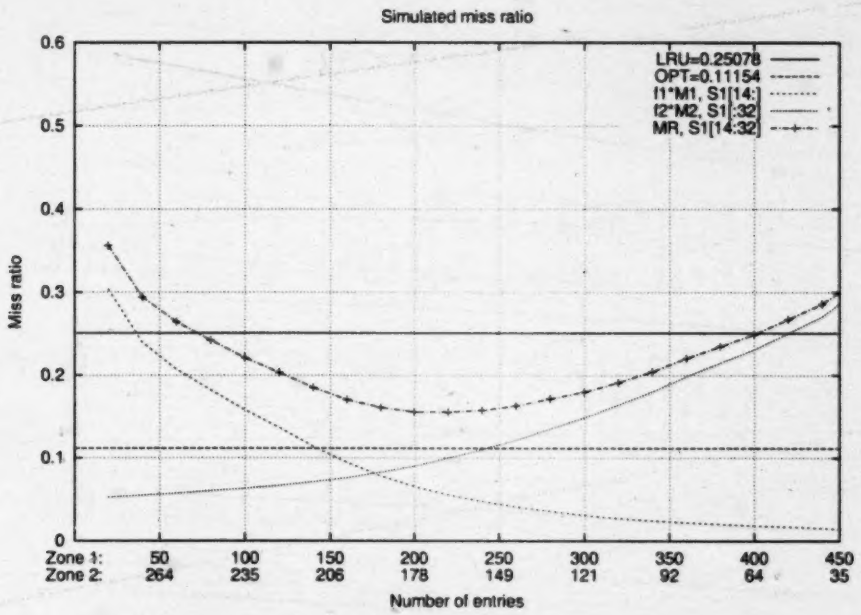


Figure B.1: Simulated miss ratio of two-zone cache for S1[14:32]

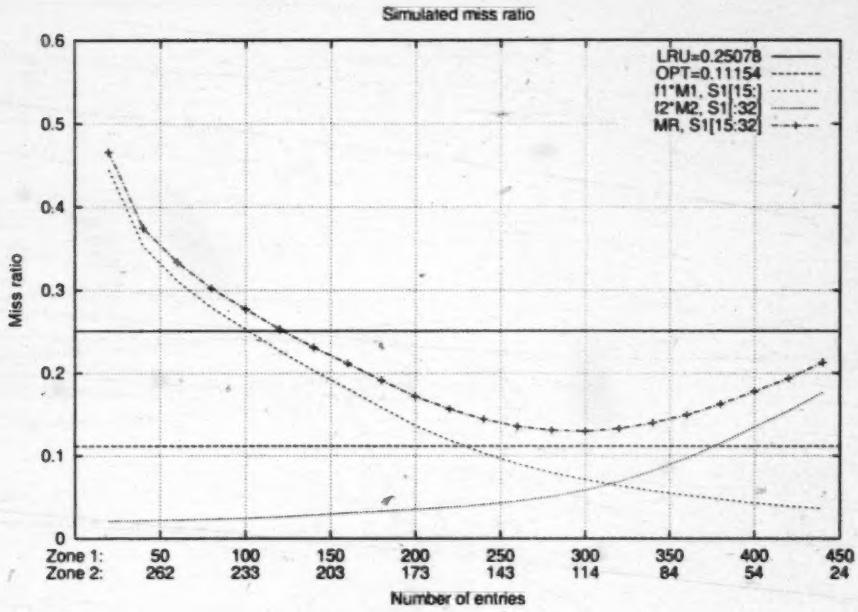


Figure B.2: Simulated miss ratio of two-zone cache for S1[15:32]

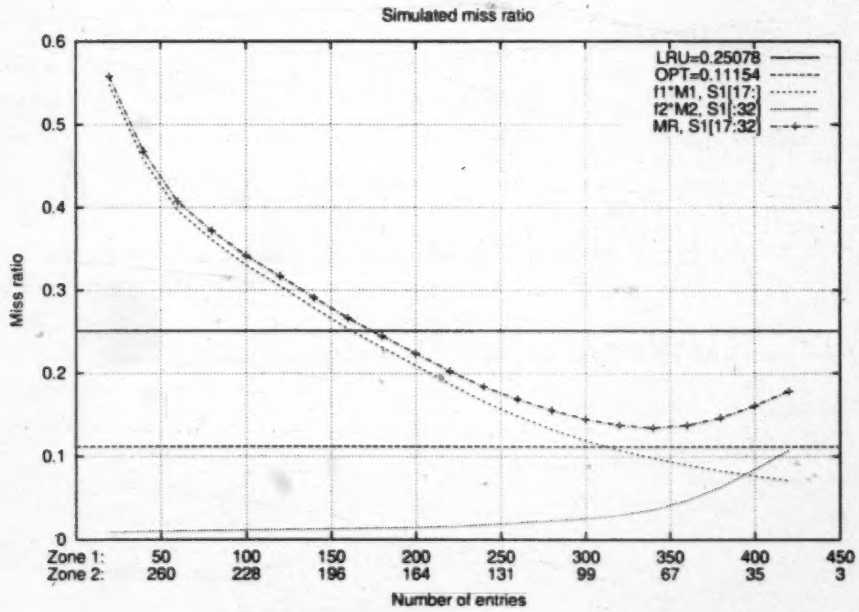


Figure B.3: Simulated miss ratio of two-zone cache for S1[17:32]

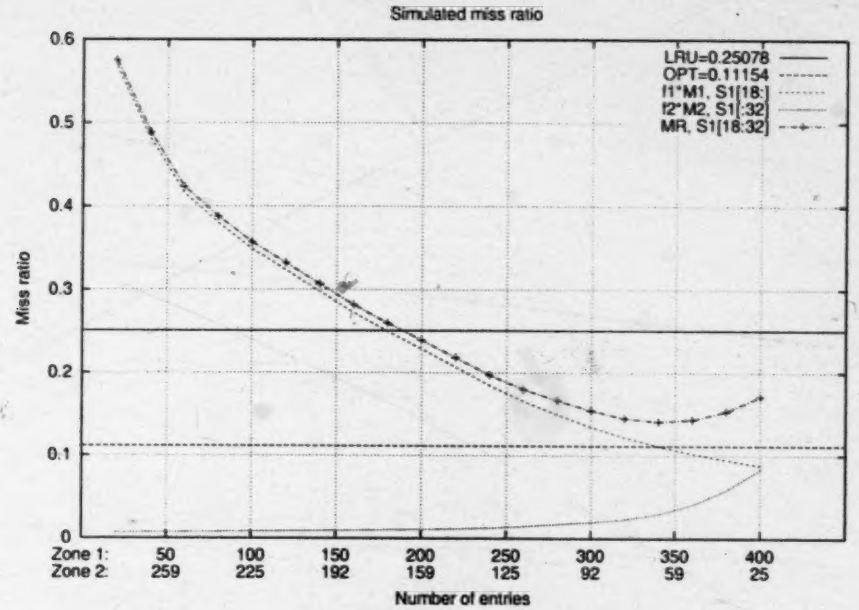


Figure B.4: Simulated miss ratio of two-zone cache for S1[18:32]

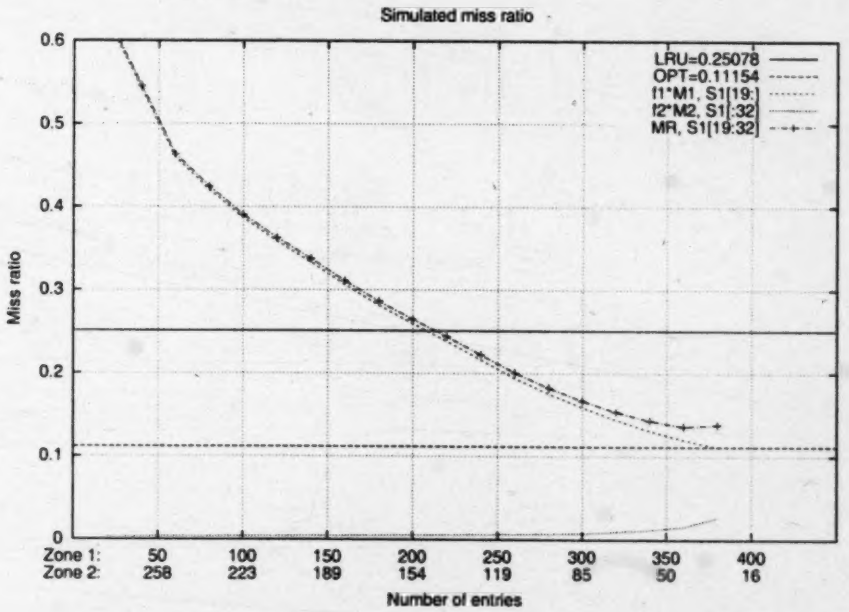


Figure B.5: Simulated miss ratio of two-zone cache for S1[19:32]

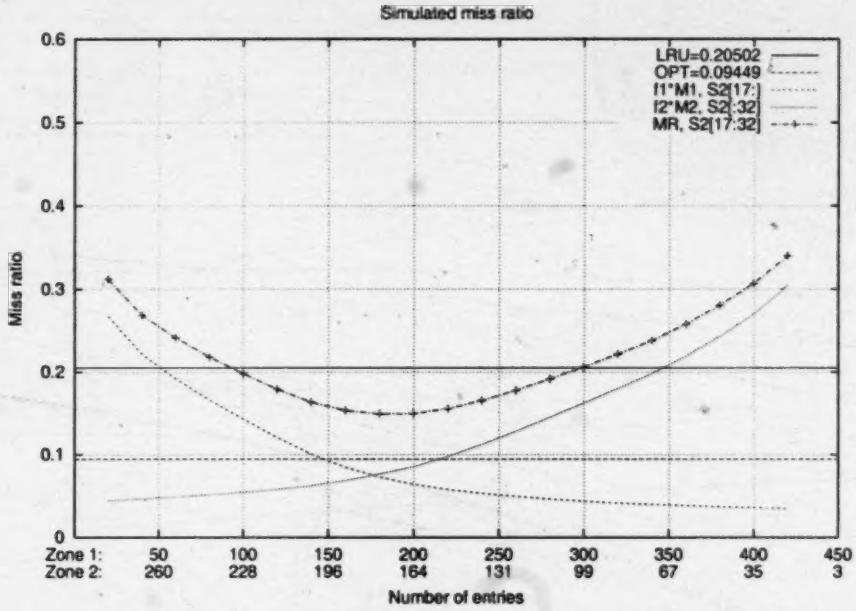


Figure B.6: Simulated miss ratio of two-zone cache for S2[17:32]

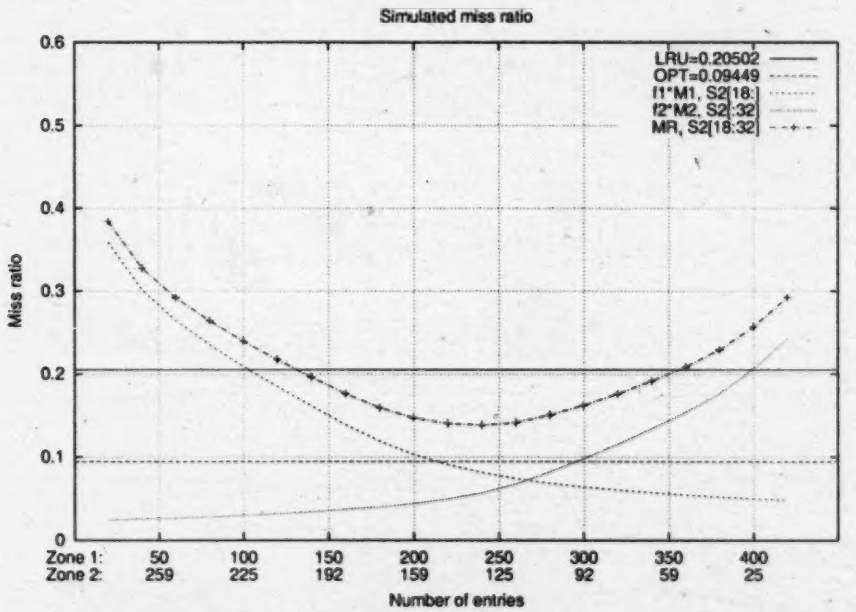


Figure B.7: Simulated miss ratio of two-zone cache for S2[18:32]

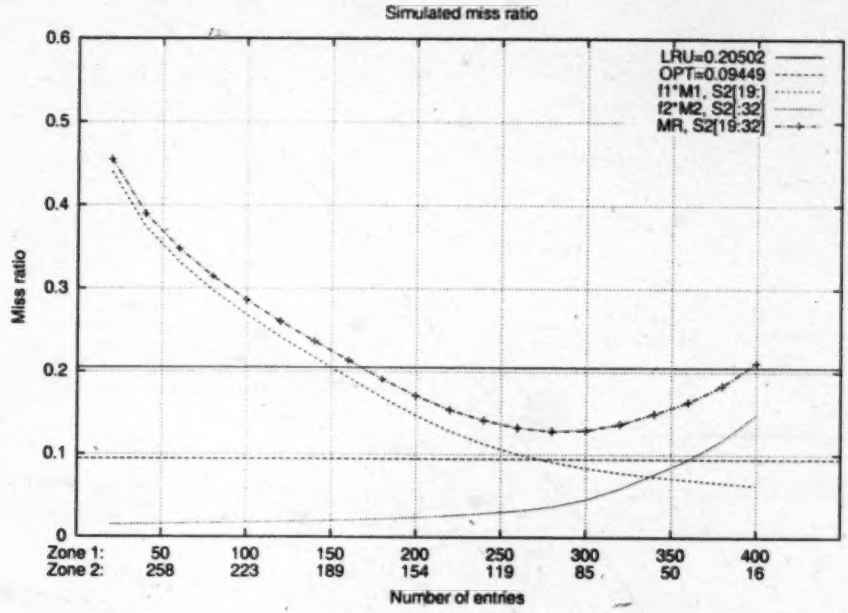


Figure B.8: Simulated miss ratio of two-zone cache for S2[19:32]

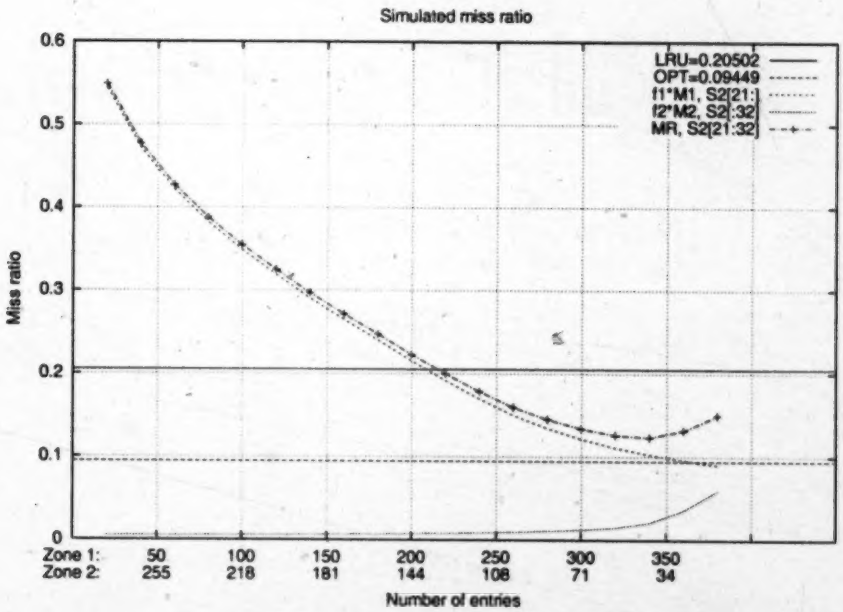


Figure B.9: Simulated miss ratio of two-zone cache for S2[21:32]

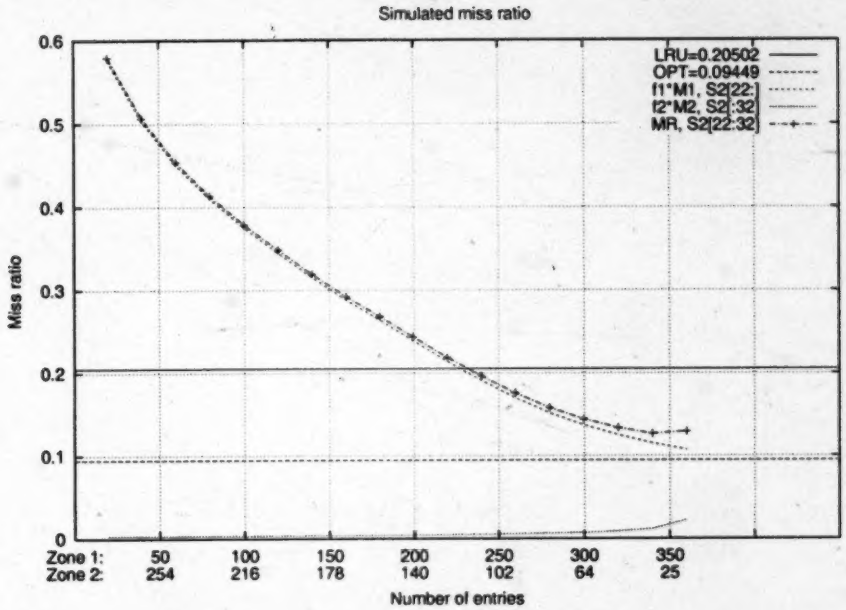


Figure B.10: Simulated miss ratio of two-zone cache for S2[22:32]

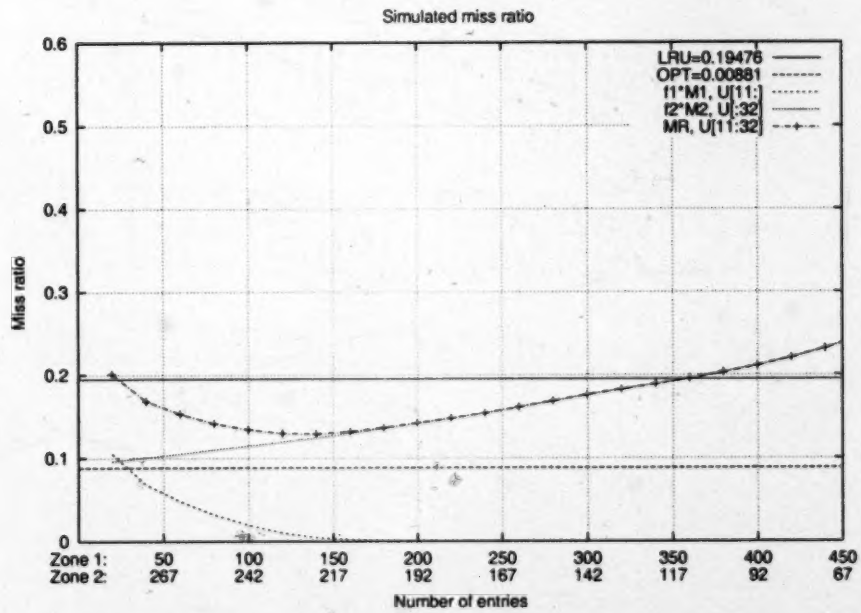


Figure B.11: Simulated miss ratio of two-zone cache for U[11:32]

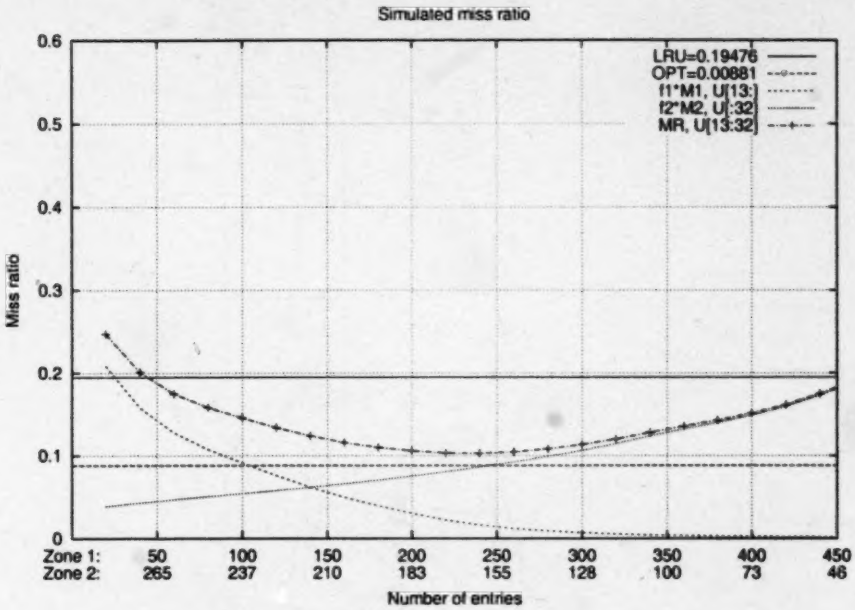


Figure B.12: Simulated miss ratio of two-zone cache for U[13:32]

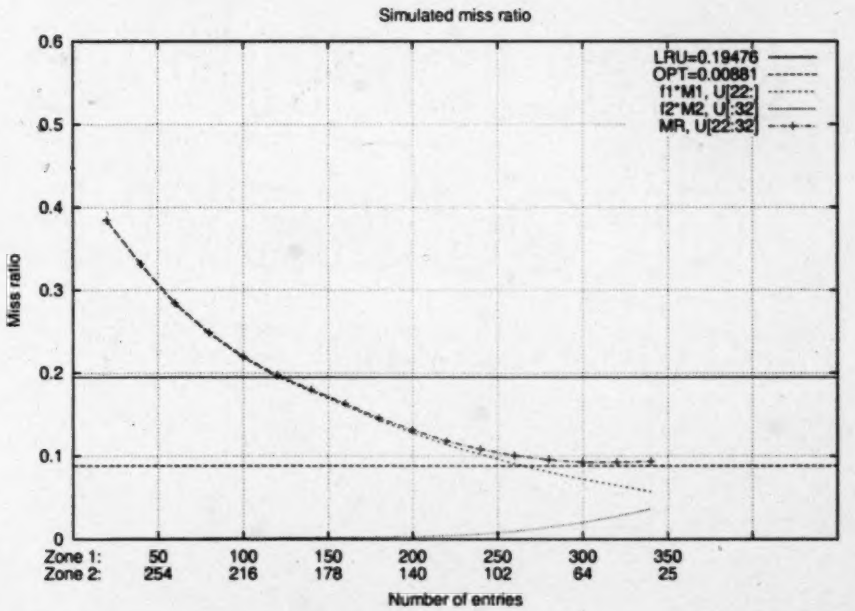


Figure B.13: Simulated miss ratio of two-zone cache for U[22:32]

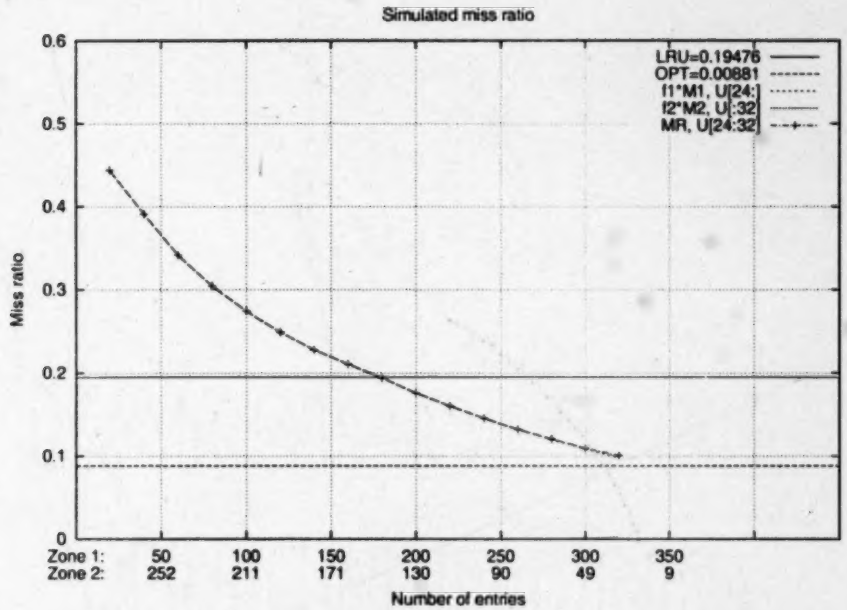


Figure B.14: Simulated miss ratio of two-zone cache for U[24:32]